การเขียนโปรแกรม

กราฟิกส์และเกมคอมพิวเตอร์ด้วยเทอร์โบปาสคาล



แล้วคุณจะพบว่า
- เขียนโปรแกรมกราฟิกส์ง่ายกว่าใช้โปรแกรมกราฟิกส์
- เขียนโปรแกรมเกมสนุกกว่าเล่นเกม

นุกูล กระจาย

ซีเอ็ด

การเขียนโปรแกรม

กราฟิกส์และเกมคอมพิวเตอร์ด้วยเทอร์โบปาสคาล

นุกูล กระจาย

บริษัท ซีเอ็ดยูเคชั่น จำกัด
SCIENCE, ENGINEERING & EDUCATION CO.,LTD.

ราคา 375 บาท

ISBN 974-509-565-6

1 3 2 – 0 0 0 – 1 5
0 6 3 3 4 3 2 1 0 9 8 7 6

จัดพิมพ์และจัดจำหน่ายโดย

บริษัท ซีเอ็ดยูเคชั่น จำกัด
SCIENCE, ENGINEERING & EDUCATION CO.,LTD.

800/43-45 ซอยตระกูลสุข ถนนอโศกดินแดง เขตห้วยขวาง
กรุงเทพฯ 10400 โทร. 2486280-9 โทรสาร 2465999

พิมพ์ที่ หจก. เอช-เอน การพิมพ์ โทร. 2112622
402/16 ซอยดอนกุศล เขตยานนาวา กทม. 10120
นายประพันธ์ จิระมงคล ผู้พิมพ์ผู้โฆษณา พ.ศ. 2536

คำนำ

หนังสือเล่มนี้มีจุดมุ่งหมายที่จะให้ผู้อ่านสามารถเขียนโปรแกรมกราฟิกส์และเกมคอมพิวเตอร์ได้ด้วยตัวเอง

ท่านอาจจะรู้จักและเคยเล่นเกมคอมพิวเตอร์ ท่านอาจจะรู้จักและเคยใช้โปรแกรมกราฟิกส์ แต่ท่านอาจจะยังไม่ทราบว่าท่านก็สามารถเขียนโปรแกรมเหล่านั้นได้

จุดมุ่งหมายของการจัดทำหนังสือเล่มนี้ มีพื้นฐานมาจากความเชื่อมั่นที่ว่า คนไทยมีความรู้ความสามารถที่จะเขียนโปรแกรมกราฟิกส์และเกมคอมพิวเตอร์ได้เหมือนกับคนชาติอื่นที่เป็นผู้นำทางด้านคอมพิวเตอร์ ซึ่งถ้าเราได้ลงมือกันอย่างจริงจัง ในไม่ช้าเราก็ไม่จำเป็นที่จะต้องคอยเสาะหาโปรแกรมสำเร็จรูปจากต่างชาติมาใช้อีกต่อไป และอาจเป็นไปได้ที่เราจะส่งโปรแกรมกราฟิกส์แบบไทย เกมคอมพิวเตอร์แบบไทย ออกไปจำหน่ายในต่างประเทศ แทนการนำเข้าแต่เพียงอย่างเดียวเช่นที่เป็นอยู่ในปัจจุบัน

ด้วยวิธีการนำเสนอเป็นขั้นตอนตามลำดับ จะทำให้ผู้อ่านเข้าใจหลักการและสามารถเขียนโปรแกรมกราฟิกส์และเกมคอมพิวเตอร์ได้ทันที พร้อมกับความรู้สึกว่าเขียนโปรแกรมกราฟิกส์ดีกว่าใช้โปรแกรมกราฟิกส์ เพราะผู้เขียนโปรแกรมจะมีอิสระไม่ต้องติดขัดกับข้อจำกัดซึ่งมักจะมีอยู่ในโปรแกรมสำเร็จรูป ส่วนผู้เขียนโปรแกรมเกมคอมพิวเตอร์ก็จะรู้สึกว่าเขียนโปรแกรมเกมให้ผู้อื่นเล่นสนุกกว่าเล่นเกมที่ผู้อื่นเขียน

เนื้อเรื่องในหนังสือมี 11 บท มีภาพประกอบ 98 ภาพ โปรแกรมตัวอย่าง 157 โปรแกรม และยูนิต 2 ยูนิต ซึ่งสรุปได้ดังต่อไปนี้

บทที่ 1 กราฟิกส์และเทอร์โบปาสคาล กล่าวถึงความรู้พื้นฐานเกี่ยวกับคอมพิวเตอร์กราฟิกส์และหลักการเขียนโปรแกรมกราฟิกส์ด้วยเทอร์โบปาสคาล มีภาพประกอบ 5 ภาพและโปรแกรมตัวอย่าง 2 โปรแกรม

บทที่ 2 การแสดงกราฟิกส์พื้นฐาน กล่าวถึงการเขียนโปรแกรมเพื่อแสดงกราฟิกส์ระดับพื้นฐานที่จอภาพ เช่น การแสดงจุด เส้นตรง เส้นโค้ง ภาพหลายเหลี่ยม วงกลม วงรี มีภาพประกอบ 2 ภาพและโปรแกรมตัวอย่าง 54 โปรแกรม

บทที่ 3 การบันทึกภาพและแสดงภาพในไฟล์ กล่าวถึงวิธีการต่าง ๆ ที่ใช้สำหรับบันทึกภาพไว้ในดิสก์ และวิธีการนำภาพที่บันทึกเอาไว้กลับมาแสดงที่จอภาพ มีภาพประกอบ 7 ภาพและโปรแกรมตัวอย่าง 10 โปรแกรม

บทที่ 4 การให้สีและลายภาพ กล่าวถึงการให้สีและลายแก่ภาพ เพื่อการแสดงภาพที่มีสีและลายแบบต่าง ๆ ที่จอภาพ มีภาพประกอบ 2 ภาพและโปรแกรมตัวอย่าง 12 โปรแกรม

บทที่ 5 การแปลงภาพกราฟิกส์ กล่าวถึงการแปลงภาพที่มีอยู่ให้มีลักษณะแตกต่างไปจากเดิม เช่น การเปลี่ยนขนาดภาพ การย้ายภาพ การหมุนภาพ การเบ้ภาพ มีภาพประกอบ 1 ภาพและโปรแกรมตัวอย่าง 17 โปรแกรม

บทที่ 6 การแสดงภาพจากไฟล์ข้อมูลภาพแบบบิตแมพ กล่าวถึงวิธีการนำข้อมูลภาพที่ใช้ในโปรแกรมอื่น มาแสดงที่จอภาพด้วยโปรแกรมที่เราเขียนขึ้นเอง ไฟล์ภาพที่นำมาเป็นตัวอย่างได้จากโปรแกรมพรินต์มาสเตอร์พลัส นิวพรินต์ชอพ และไฟล์ภาพที่ได้จากการสแกนด้วยสแกนเนอร์ มีภาพประกอบ 5 ภาพและโปรแกรมตัวอย่าง 11 โปรแกรม

บทที่ 7 การแปลงภาพบนจอภาพ กล่าวถึงการเปลี่ยนภาพที่แสดงอยู่บนจอภาพ ให้มีลักษณะแตกต่างไปจากเดิม เช่น การเปลี่ยนขนาด การกลับภาพ การกลับสี การเบ้ภาพ มีภาพประกอบ 15 ภาพและโปรแกรมตัวอย่าง 12 โปรแกรม

บทที่ 8 การพิมพ์ภาพกราฟิกส์ กล่าวถึงการนำภาพที่ปรากฏอยู่บนจอภาพมาพิมพ์ที่เครื่องพิมพ์เพื่อให้ภาพเกิดขึ้นบนกระดาษ โดยมีทั้งการพิมพ์ภาพแบบธรรมดา ซึ่งเหมือนกับภาพที่ปรากฏบนจอภาพและการพิมพ์ให้แตกต่างกัน เช่น ภาพย่อ ภาพขยาย ภาพกลับสี ภาพเบ้ มีภาพประกอบ 21 ภาพและโปรแกรมตัวอย่าง 16 โปรแกรม

บทที่ 9 การแสดงและพิมพ์กราฟจากข้อมูลที่กำหนดให้ กล่าวถึงโปรแกรมที่เขียนขึ้นในลักษณะของโปรแกรมสำเร็จรูป เพื่อใช้สำหรับนำข้อมูลซึ่งเป็นตัวเลข และตัวอักษร มาแสดงเป็นกราฟ โดยสามารถแสดงได้ทั้งที่จอภาพและพิมพ์บนกระดาษ กราฟที่แสดงได้มี 4 ชนิดคือ กราฟจุด กราฟเส้น กราฟแท่งและกราฟพาย มีภาพประกอบ 8 ภาพและโปรแกรมตัวอย่าง 1 โปรแกรม

บทที่ 10 การสร้างและแสดงอักษรกราฟิกส์ กล่าวถึงการนำไฟล์อักษรจากโปรแกรม RW และ CW มาแสดงที่จอภาพ หลักการสร้างอักษรกราฟิกส์และโปรแกรมสำเร็จรูปสำหรับใช้สร้างและแสดงอักษรกราฟิกส์ มีภาพประกอบ 15 ภาพและโปรแกรมตัวอย่าง 14 โปรแกรม

บทที่ 11 โปรแกรมเกมคอมพิวเตอร์ กล่าวถึงหลักการสร้างเกมคอมพิวเตอร์และตัวอย่างโปรแกรมเกมรวม 8 เกม มีภาพประกอบ 17 ภาพและโปรแกรมตัวอย่าง 8 โปรแกรม

นอกจากนี้ ในภาคผนวก ก. ยังมีคำแนะนำเพิ่มเติมเกี่ยวกับวิธีการนำไดรเวอร์ และฟอนต์ต่าง ๆ ที่มีอยู่ในเทอร์โบปาสคาล มารวมอยู่ในโปรแกรมที่เราเขียนขึ้นซึ่งเมื่อคอมไพล์ให้เป็นไฟล์ชนิด EXE แล้วจะสามารถนำโปรแกรมที่อยู่ในไฟล์นั้นไปรันกับคอมพิวเตอร์ที่มีระบบกราฟิกส์ชนิดใดก็ได้โดยไม่ต้องใช้ไฟล์ไดรเวอร์จากเทอร์โบปาสคาลอีกเลย ในส่วนนี้มียูนิตรวม 2 ยูนิต

เนื่องจากเรื่องราวในหนังสือเล่มนี้เป็นการนำความรู้ความสามารถทางด้านการเขียนโปรแกรมปาสคาลมาประยุกต์ใช้ และเพื่อไม่ให้หนังสือมีความหนามากเกินไป จึงไม่ได้กล่าวถึงหลักการเขียนโปรแกรมปาสคาลเบื้องต้นไว้ในที่นี้ สำหรับท่านที่ต้องการทราบความรู้พื้นฐานเกี่ยวกับการเขียนโปรแกรมปาสคาลจะหาอ่านได้จากหนังสือ*การเขียนโปรแกรมและประมวลผลข้อมูลด้วยเทอร์โบปาสคาล* เขียนโดย *นุกูล กระจาย* จัดพิมพ์จำหน่ายโดย *บริษัท ซีเอ็ดยูเคชั่น* จำกัด

มีข้อสังเกตเกี่ยวกับโปรแกรมที่นำมาเป็นตัวอย่างในหนังสือเล่มนี้คือ ตามปกติภาษาปาสคาลจะเน้นการเขียนโปรแกรมให้เป็นส่วนย่อยในลักษณะของโพรซีเยอร์และฟังก์ชัน แต่ในหนังสือเล่มนี้ได้นำเสนอในลักษณะของโปรแกรมเล็ก ๆ ทั้งนี้ก็เพื่อที่จะแสดงผลของโปรแกรมให้เห็นได้อย่างรวดเร็ว อย่างไรก็ตาม โปรแกรมตัวอย่างเหล่านี้สามารถปรับปรุงให้เป็นโปรแกรมระดับใช้งานซึ่งเป็นโปรแกรมขนาดใหญ่ได้ โดยการนำโพรซีเยอร์และฟังก์ชันที่มีอยู่ในโปรแกรมต่าง ๆ ที่เกี่ยวข้องกันมารวมกัน

โปรแกรมทุกโปรแกรมในหนังสือเล่มนี้ ใช้ได้กับเทอร์โบปาสคาลตั้งแต่เวอร์ชัน 4.0 ถึงเวอร์ชัน 6.0 ทุกโปรแกรมได้ผ่านการทดสอบเรียบร้อยแล้ว โดยทำการทดสอบทั้งในคอมพิวเตอร์ที่มีจอภาพสีระบบวีจีเอ (VGA) และในคอมพิวเตอร์ที่มีจอภาพโมโนโครมระบบเฮอร์คิวลีส (Hercules) เนื่องจากระบบจอภาพทั้งสองระบบนี้มีรายละเอียดแตกต่างกัน ดังนั้นบางโปรแกรมที่เขียนไว้ในลักษณะกลาง ๆ จึงอาจจะทำงานได้สมบูรณ์ในระบบวีจีเอ แต่จะผิดเพี้ยนไปบ้างในระบบเฮอร์คิวลีส และบางโปรแกรมอาจจะทำงานได้สมบูรณ์ในระบบเฮอร์คิวลีส แต่จะผิดเพี้ยนไปบ้างในระบบวีจีเอ อย่างไรก็ดี เชื่อว่าผู้อ่านทุกท่านสามารถจะปรับปรุงโปรแกรมดังกล่าวให้สมบูรณ์ได้โดยไม่ยาก

แม้จะได้จัดทำหนังสือนี้ด้วยความตั้งใจและละเอียดรอบคอบแล้วก็ตาม ความบกพร่องบางประการก็อาจจะยังคงหลงเหลืออยู่ ผู้เขียนยินดีที่จะรับทราบในทุกสิ่งที่ผิดพลาด เพื่อจะได้แก้ไขให้ถูกต้องต่อไป และขอขอบคุณท่านผู้อ่านที่จะได้สละเวลาเพื่อแสดงความคิดเห็นให้ผู้เขียนได้ทราบบ้าง

ขอให้ผู้อ่านทุกท่านได้บรรลุผลตามวัตถุประสงค์ของหนังสือเล่มนี้อย่างสมบูรณ์โดยทั่วกัน

นุกูล กระจาย

คณะวิทยาศาสตร์

มหาวิทยาลัยศรีนครินทรวิโรฒ ประสานมิตร

ดิสก์ 157 โปรแกรม

เนื่องจากโปรแกรมที่มีในหนังสือเล่มนี้ ส่วนใหญ่เป็นโปรแกรมที่มีความยาวมาก ผู้อ่านบางท่านอาจไม่สะดวกที่จะพิมพ์โปรแกรมด้วยตัวเอง ผู้เขียนจึงได้จัดเตรียมโปรแกรมทั้งหมดที่มีอยู่ในหนังสือเล่มนี้ รวมทั้งไฟล์ภาพที่จำเป็นต้องใช้บางไฟล์ โดยบันทึกไว้ในดิสก์ขนาด $5^1/4$ นิ้ว (DS/DD)

ท่านผู้อ่านที่มีความสนใจดิสก์ 157 โปรแกรม จะสามารถซื้อได้ในราคา 150 บาท หรือสั่งซื้อทางไปรษณีย์โดยส่งธนาณัติสั่งจ่าย ปทฝ.ดินแดง หรือเช็คธนาคารในเขตกรุงเทพมหานคร ในนามบริษัท ซีเอ็ดยูเคชั่น จำกัด จำนวน 165 บาท (รวมค่าจัดส่ง 15 บาท) ที่อยู่ของบริษัทคือ

บริษัท ซีเอ็ดยูเคชั่น จำกัด
800/43-45 ซ. ตระกูลสุข ถ. อโศกดินแดง
เขตห้วยขวาง กทม. 10400 โทร. 2486280-9

หรือซื้อได้ที่ *ซีเอ็ดบุ๊คเซ็นเตอร์* ทุกสาขาคือ
ฟอร์จูนทาวน์ (เยาฮัน) ชั้นใต้ดิน
สีลมคอมเพล็กซ์ ชั้น 4
หลักสี่พลาซ่า ชั้น 3
อิมพีเรียลเวิลด์ สำโรง ชั้นใต้ดิน
เซ็นทรัลซิตี้ บางนา ชั้น 3
เซ็นทรัลปิ่นเกล้า ชั้น 3

สารบัญ

1

กราฟิกส์และเทอร์โบปาสคาล

ในบทนี้จะกล่าวถึงความรู้เบื้องต้นเกี่ยวกับกราฟิกส์ และหลักการเขียนโปรแกรมเพื่อแสดงกราฟิกส์ด้วยเทอร์โบปาสคาล

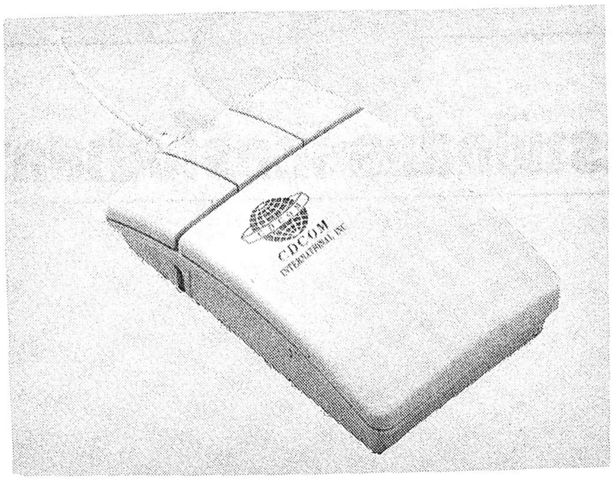
1.1 ความหมายของคอมพิวเตอร์กราฟิกส์

คอมพิวเตอร์กราฟิกส์หมายถึง การใช้คอมพิวเตอร์ดำเนินการเกี่ยวกับภาพ ซึ่งได้แก่การสร้างภาพ การแสดงภาพ การบันทึกภาพ การรับภาพ และการพิมพ์ภาพ

1.2 ส่วนประกอบของคอมพิวเตอร์กราฟิกส์

การดำเนินการเกี่ยวกับภาพด้วยคอมพิวเตอร์จะต้องอาศัยส่วนประกอบ 3 ประการคือ

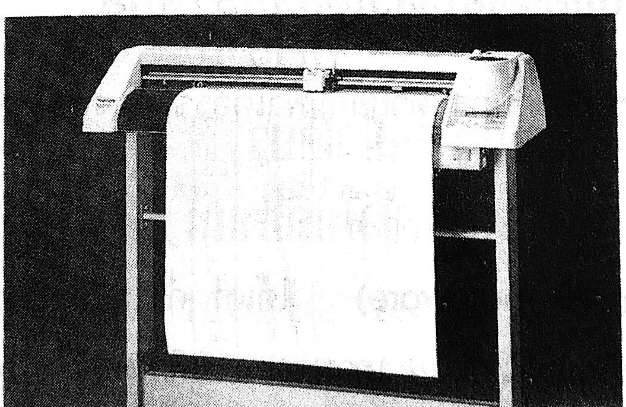
1.2.1 ฮาร์ดแวร์ (hardware) ได้แก่ ตัวเครื่องคอมพิวเตอร์และอุปกรณ์ประกอบต่างๆ สำหรับทำหน้าที่แสดงภาพจากคอมพิวเตอร์ รับภาพเข้ามาในคอมพิวเตอร์ และบันทึกภาพจากคอมพิวเตอร์ ตัวอย่างอุปกรณ์แสดงภาพได้แก่ จอภาพ เครื่องพิมพ์ พล็อตเตอร์ (plotter) อุปกรณ์รับภาพได้แก่ คีย์บอร์ด เมาส์(mouse) สแกนเนอร์ (scanner) อุปกรณ์บันทึกภาพ ได้แก่ ดิสก์



(ก) เมาส์



(ข) สแกนเนอร์



(ค) พล็อตเตอร์

รูปที่ 1.1 อุปกรณ์คอมพิวเตอร์บางชนิดที่ใช้ในงานคอมพิวเตอร์กราฟิกส์

1.2.2. ซอฟต์แวร์ (software) ได้แก่ โปรแกรมคอมพิวเตอร์ซึ่งสั่งให้คอมพิวเตอร์ดำเนินการเกี่ยวกับภาพ โปรแกรมคอมพิวเตอร์แบ่งได้เป็น 2 ชนิดคือ โปรแกรมสำเร็จรูป (package) และโปรแกรมที่เราเขียนขึ้นเอง

โปรแกรมสำเร็จรูป เป็นโปรแกรมที่สร้างขึ้นโดยผู้เชี่ยวชาญทางด้านคอมพิวเตอร์กราฟิกส์ เพื่อให้ผู้ใช้โปรแกรมสามารถใช้คอมพิวเตอร์ดำเนินการเกี่ยวกับภาพได้อย่างสะดวกรวดเร็ว ปัจจุบันมีผู้ผลิตโปรแกรมสำเร็จรูปทางด้านกราฟิกส์ออกจำหน่ายเป็นจำนวนมาก

โปรแกรมกราฟิกส์ที่เราเขียนขึ้นเอง เป็นโปรแกรมที่เขียนขึ้นด้วยภาษาคอมพิวเตอร์ เช่น เบสิก ฟอร์แทรน ปาสคาล และอื่นๆ โดยเขียนด้วยคำและหลักการของภาษานั้น เพื่อให้คอมพิวเตอร์ดำเนินการเกี่ยวกับกราฟิกส์ตามที่เราต้องการ

เนื่องจากคอมพิวเตอร์กราฟิกส์เป็นงานที่สิ้นเปลืองเวลา ทั้งยังต้องใช้กำลังสมองและกำลังกายเป็นอย่างมาก เพราะฉะนั้น เราจึงต้องพิจารณาว่าโปรแกรมกราฟิกส์ที่จะนำมาใช้งาน ควรจะเป็นโปรแกรมสำเร็จรูป โปรแกรมที่เขียนขึ้นเอง หรือใช้โปรแกรมทั้งสองชนิดร่วมกัน แนวทางการตัดสินใจเลือกใช้โปรแกรมกราฟิกส์ อาจจะพิจารณาได้จากข้อมูลต่างๆ ดังต่อไปนี้

1. โปรแกรมสำเร็จรูปสามารถใช้งานได้ทันที โดยเสียเวลาศึกษาวิธีการใช้โปรแกรมเพียงเล็กน้อยเท่านั้น โปรแกรมสำเร็จรูปจึงเหมาะกับงานเร่งด่วน และใช้ในการศึกษาของผู้เริ่มต้น

2. โปรแกรมสำเร็จรูปแต่ละโปรแกรม มีจุดมุ่งหมายของการใช้งานแตกต่างกัน เช่น บางโปรแกรมเน้นทางด้านการพิมพ์ภาพ บางโปรแกรมเน้นทางด้านการพิมพ์ตัวอักษร ดังนั้นก่อนที่จะตัดสินใจเลือกโปรแกรมมาใช้งานจึงต้องทำการศึกษาและอาจจะต้องทดลองใช้โปรแกรมนั้นดูก่อน ซึ่งเป็นขั้นตอนที่สิ้นเปลืองเวลา และถ้าเลือกโปรแกรมไม่เหมาะสมก็จะยิ่งทำให้เสียทั้งเวลาและทรัพย์สินไปโดยเปล่าประโยชน์

3. มีความเป็นไปได้สูงมาก ที่โปรแกรมสำเร็จรูปเพียงโปรแกรมเดียว จะไม่สามารถทำงานให้ตรงกับความต้องการของเราได้ครบถ้วน เช่น โปรแกรมจากต่าง

ประเทศใช้สร้างอักษรไทยไม่ได้ สร้างเสียงที่เราต้องการไม่ได้ จึงอาจจำเป็นจะต้องใช้โปรแกรมร่วมกันครั้งละหลายโปรแกรม กรณีที่มีปัญหาดังกล่าวนี้ การเขียนโปรแกรมขึ้นเองจึงเป็นวิธีแก้ปัญหาที่ดีที่สุด เพราะเราสามารถเขียนโปรแกรมให้ทำทุกอย่างได้ตามที่เราต้องการ

4. ในระยะยาว การใช้โปรแกรมสำเร็จรูปทำให้สิ้นเปลืองมากกว่า เนื่องจากจะต้องหาซื้อโปรแกรมรุ่นใหม่มาใช้แทนโปรแกรมรุ่นเก่าอยู่เสมอ โปรแกรมที่เปลี่ยนรุ่นเร็วจะมีผลให้ต้องมีค่าใช้จ่ายเพิ่มขึ้นเร็วกว่าด้วย ต่างกับโปรแกรมที่เขียนขึ้นเองซึ่งเราสามารถปรับปรุงเพิ่มเติมให้เหมาะสมกับความต้องการ และสมัยนิยมด้วยค่าใช้จ่ายที่ต่ำกว่ามาก

5. ผู้ผลิตโปรแกรมสำเร็จรูปมีฐานะเป็นผู้ขายหรือผู้รับเงิน ส่วนผู้ใช้โปรแกรมสำเร็จรูปมีฐานะเป็นผู้ซื้อหรือผู้จ่ายเงิน ถ้าผู้ใช้โปรแกรมสำเร็จรูปไม่พยายามพัฒนาความรู้ความสามารถก็คงจะต้องเป็นผู้ซื้อตลอดไป วิธีการที่เหมาะสมสำหรับระยะยาวก็คือผู้ใช้โปรแกรมสำเร็จรูปในวันนี้ ควรจะพยายามศึกษาและสร้างโปรแกรมขึ้นใช้เองให้ได้เพื่อให้สามารถพึ่งตนเองได้ในวันข้างหน้า และอาจจะเปลี่ยนเป็นผู้ขายโปรแกรมสำเร็จรูปในอนาคต

6. การเขียนโปรแกรมขึ้นใช้เอง ทำให้เราเกิดความเข้าใจเกี่ยวกับฮาร์ด-แวร์ และซอฟต์แวร์ของคอมพิวเตอร์ ที่ใช้งานด้านกราฟิกส์ได้ดีขึ้น ซึ่งเป็นผลให้เรามีความมั่นใจและภูมิใจในความรู้ความสามารถของตนเอง

1.2.3 บุคลากร คือกลุ่มคนที่มีส่วนเกี่ยวข้องกับคอมพิวเตอร์กราฟิกส์ ซึ่งอาจแบ่งเป็น 2 กลุ่มคือ ผู้ผลิตโปรแกรม และผู้ใช้โปรแกรม

ผู้ผลิตโปรแกรมทำหน้าที่เกี่ยวกับ การออกแบบโปรแกรม เขียนโปรแกรม ทด-สอบการทำงานของโปรแกรม และเขียนคู่มือการใช้โปรแกรม

ผู้ใช้โปรแกรม มีหน้าที่ศึกษาวิธีการใช้โปรแกรมและใช้โปรแกรม

จะเห็นได้ว่าผู้ผลิตโปรแกรมคือผู้นำที่มีผู้ใช้โปรแกรมเป็นผู้ตาม

1.3 ประโยชน์ของคอมพิวเตอร์กราฟิกส์

ได้มีการนำคอมพิวเตอร์กราฟิกส์มาใช้ในงานต่างๆ อย่างกว้างขวาง เช่น

1. ใช้แสดงผลงานด้วยภาพแทนการแสดงด้วยข้อความ ซึ่งช่วยให้เข้าใจได้ง่าย และน่าสนใจมากกว่า หลายหน่วยงานเลือกใช้วิธีนี้สำหรับการแนะนำหน่วยงาน เสนอโครงการและแสดงผลงาน

2. ใช้แสดงแผนที่ แผนผัง และภาพของสิ่งต่างๆ ซึ่งภาพเหล่านี้ไม่สามารถแสดงในลักษณะอื่นได้ นอกจากการแสดงด้วยภาพเท่านั้น

3. ใช้ในการออกแบบทางด้านต่างๆ เช่น ออกแบบบ้าน รถยนต์ เครื่องจักร เครื่องแต่งกาย การแต่งหน้า และเครื่องมือเครื่องใช้อื่นๆ ซึ่งสามารถทำได้รวดเร็ว สวยงามและประหยัดค่าใช้จ่าย โดยเฉพาะงานออกแบบที่มีการเปลี่ยนแปลงรายละเอียดเพื่อเปรียบเทียบหาแบบที่เหมาะสมที่สุด การใช้คอมพิวเตอร์กราฟิกส์จะช่วยให้เกิดความสะดวก และทำได้รวดเร็วมาก

4. ได้มีการนำคอมพิวเตอร์กราฟิกส์มาช่วยทางด้านการเรียนการสอน โดยเฉพาะในวิชาที่ต้องใช้ภาพ แผนผัง หรือแผนที่ประกอบ บทเรียนคอมพิวเตอร์สามารถแสดงภาพส่วนประกอบและการทำงานของเครื่องยนต์ หรือเครื่องมือที่มีความสลับซับซ้อนให้เห็นได้ง่ายขึ้น

5. คอมพิวเตอร์กราฟิกส์ถูกนำมาใช้ในการจำลองสถานการณ์ เพื่อหาคำตอบว่า ถ้าสถานการณ์เป็นอย่างนี้แล้วจะเกิดอะไรขึ้น เช่น ผู้ผลิตรถยนต์ใช้ทดสอบว่าถ้ารถยนต์รุ่นนี้พุ่งเข้าชนกำแพงด้วยความเร็ว 40 กิโลเมตรต่อชั่วโมง จะเกิดความเสียหายที่บริเวณไหน ผู้โดยสารจะเป็นอย่างไร การจำลองสถานการณ์ด้วยคอมพิวเตอร์ช่วยให้ทราบผลได้รวดเร็ว ประหยัดค่าใช้จ่ายและไม่ทำให้เกิดอันตราย

6. คอมพิวเตอร์กราฟิกส์สามารถนำมาสร้างเป็นภาพนิ่ง ภาพสไลด์ ภาพยนตร์ และรายการวิดีโอ ได้มีภาพยนตร์แนววิทยาศาสตร์หลายเรื่องใช้คอมพิวเตอร์กราฟิกส์สร้างฉากและตัวละคร ซึ่งทำให้ดูสมจริงได้ดีกว่าการสร้างด้วยวิธีอื่น

7. คอมพิวเตอร์กราฟิกส์ที่มีผู้รู้จักและนิยมใช้กันมากคงจะได้แก่ เกมคอมพิวเตอร์ ในปัจจุบันนี้คงมีคนจำนวนน้อยเท่านั้นที่ไม่เคยเห็นหรือรู้จักเกมคอมพิวเตอร์ ส่วนประกอบหนึ่งที่ทำให้เกมสนุกและน่าสนใจก็คือ ภาพของฉากและตัวละครในเกมซึ่งสร้างโดยคอมพิวเตอร์กราฟิกส์

1.4 พิกเซล

พิกเซล(pixel) เป็นส่วนเล็กที่สุดที่สามารถแสดงให้เห็นที่จอภาพและที่อุปกรณ์แสดงผลประเภทกราฟิกส์อื่นๆ ของระบบคอมพิวเตอร์ พิกเซลมีความสำคัญต่อการสร้างกราฟิกส์ของคอมพิวเตอร์มาก เพราะทุกๆ ส่วนของกราฟิกส์เช่น จุด เส้น แบบลายและสีของภาพ ล้วนเริ่มมาจากพิกเซลทั้งสิ้น พิกเซลหนึ่งๆ อาจจะมีขนาดความเข้มและสีแตกต่างกันได้ ในอุปกรณ์แสดงกราฟิกส์อย่างง่ายอาจทำได้เพียงแสดงหรือไม่แสดงพิกเซล

1.5 รีโซลูชัน

รีโซลูชัน (resolution) หมายถึงรายละเอียดที่อุปกรณ์แสดงกราฟิกส์ชนิดหนึ่งมีอยู่ ค่ารีโซลูชันมักระบุเป็นจำนวนพิกเซลในแนวนอนคือแนวแกน X และจำนวนพิกเซลในแนวตั้งคือแนวแกน Y ดังนั้นรีโซลูชัน 720 x 348 จึงหมายความว่า อุปกรณ์แสดงกราฟิกส์ชนิดนี้สามารถแสดงพิกเซลในแนวนอนได้ไม่เกิน 720 พิกเซล และแสดงพิกเซลในแนวตั้งได้ไม่เกิน 348 พิกเซล ผู้ผลิตอุปกรณ์แสดงกราฟิกส์บางรายจะระบุค่ารีโซลูชันเป็นระดับสูง (high resolution) ปานกลาง (medium resolution) และระดับต่ำ (low resolution) โดยพิจารณาจากจำนวนพิกเซลในแนวนอนเพียงอย่างเดียว ซึ่งมีหลักว่า ถ้าค่าน้อยกว่า 128 เป็นระดับต่ำ ค่าระหว่าง 128 ถึง 512 เป็นระดับปานกลาง ค่าสูงกว่า 512 เป็นระดับสูง สำหรับจอภาพขนาดปกติ ถ้ามีค่ารีโซลูชันมากกว่า 1500 ตาจะมองเกือบไม่เห็นแต่ละพิกเซลคือจะมองเห็นเป็นภาพที่มีความละเอียดคมชัดสูงมาก คอมพิวเตอร์กราฟิกส์ที่ใช้กับฟิล์มถ่ายรูปในระดับมืออาชีพจะต้องใช้ค่ารีโซลูชันสูงถึง 3000

1.6 ภาพแบบบิตแมพและภาพแบบเวกเตอร์

ภาพที่สร้างด้วยคอมพิวเตอร์ มีวิธีการสร้าง 2 แบบคือ แบบบิตแมพ (bit mapped) และแบบเวกเตอร์ (vector) หรือสโตรก (stroked) แต่ละแบบมีวิธีการสร้างภาพดังต่อไปนี้

1.6.1 ภาพแบบบิตแมพ สร้างขึ้นโดยการจัดให้มีพิกเซลอยู่ในตำแหน่งต่างๆ จนสามารถมองเห็นเป็นภาพที่ต้องการ จะเห็นได้ว่าภาพแบบบิตแมพ มีวิธีการสร้างคล้ายกับการเขียนจุดลงบนกระดาษกราฟ ซึ่งถ้าเขียนจุดที่ตำแหน่งต่างๆ ได้เหมาะสม เราจะมองเห็นเป็นภาพ

1.6.2 ภาพแบบเวกเตอร์ สร้างขึ้นโดยการใช้คำสั่งลากเส้นตรงเชื่อมต่อจุดต่างๆ ในทิศทางและตำแหน่งที่กำหนดเอาไว้ เช่น การสร้างรูปสามเหลี่ยมจะมีคำสั่งดังนี้

เลื่อนไปที่จุด (0,0)
ลากเส้นไปที่จุด (300,0)
ลากเส้นไปที่จุด (150,200)
ลากเส้นไปที่จุด (0,0)

เปรียบเทียบคุณสมบัติของภาพแบบบิตแมพและภาพแบบเวกเตอร์ ในด้านความเร็วของการแสดงภาพที่จอภาพและความสามารถในการเปลี่ยนขนาดภาพจะได้ผลดังนี้

1. ภาพแบบบิตแมพสามารถแสดงให้เห็นที่จอภาพได้เร็วกว่าภาพแบบเวกเตอร์ เช่น การแสดงภาพแบบบิตแมพขนาด 1000 ไบต์ จะทำโดยการใช้คำสั่งย้ายข้อมูลขนาด 1000 ไบต์ จากหน่วยความจำที่เก็บภาพไปยังหน่วยความจำของจอภาพ (คือ video display buffer) ภาพนั้นก็จะปรากฏบนจอภาพทันที การแสดงภาพแบบเวกเตอร์คอมพิวเตอร์จะใช้เวลามากกว่า เนื่องจากคอมพิวเตอร์ต้องทำตามคำสั่งที่มีจำนวนมากกว่า

2. การเปลี่ยนแปลงขนาดภาพให้โตขึ้นหรือเล็กลงกว่าภาพเดิม กรณีภาพแบบบิตแมพจะทำได้ไม่มาก นอกจากนั้นยังอาจจะทำให้ลักษณะของภาพผิดเพี้ยนไปจากเดิมด้วย เนื่องจากการเปลี่ยนแปลงขนาดภาพทำโดยวิธีการเพิ่มหรือลดพิกเซลจากที่มีอยู่เดิม ภาพที่ขยายโตขึ้นจะมองเห็นเป็นตารางสี่เหลี่ยมเรียงต่อกัน ทำให้ขาดความสวยงาม แต่ภาพแบบเวกเตอร์จะสามารถย่อและขยายขนาดได้มากกว่า โดยสัดส่วนและลักษณะของภาพยังคล้ายเดิม ยิ่งกว่านั้นเรายังสามารถขยายเฉพาะความกว้างหรือความสูง เพื่อให้มองเห็นเป็นภาพผอมหรืออ้วนกว่าภาพเดิมได้ด้วย

1.7 ระบบกราฟิกส์

คอมพิวเตอร์ในกลุ่ม IBM PC XT AT หรือเครื่องที่ทำงานเหมือนกันจะแบ่งการแสดงผลที่จอภาพเป็น 2 แบบ หรือ 2 โหมด (mode) คือ เท็กซ์โหมด (text mode) และกราฟิกส์โหมด (graphics mode) แต่ละโหมดมีรายละเอียดดังนี้

1.7.1 เท็กซ์โหมด คอมพิวเตอร์ทุกเครื่องจะแสดงผลในโหมดนี้ได้ โดยการนำตัวอักษร ตัวเลข และเครื่องหมายต่างๆ ที่มีอยู่ในหน่วยความจำของคอมพิวเตอร์มาแสดงที่จอภาพตามคำสั่ง แต่เนื่องจากตัวอักษร ตัวเลขและเครื่องหมายที่มีอยู่ ถูกกำหนดรูปร่างไว้แน่นอนแล้ว และมีจำนวนจำกัด จึงไม่สามารถนำมาประกอบกันให้เกิดเป็นภาพต่างๆ ที่ถูกต้องสวยงามได้เท่าที่ควร

1.7.2 กราฟิกส์โหมด เพื่อให้คอมพิวเตอร์สามารถแสดงผลเป็นพิกเซลได้จำนวนมากจึงได้มีการสร้างวงจรอิเล็กทรอนิกส์ เพื่อใช้สำหรับควบคุมการแสดงผลที่จอภาพ ซึ่งนิยมเรียกกันว่า *ระบบกราฟิกส์* ระบบกราฟิกส์มีหลายชนิด เช่น ซีจีเอ (CGA) อีจีเอ (EGA) วีจีเอ (VGA) เฮอร์คิวลีส (Hercules) ซึ่งแต่ละชนิดจะมีคุณสมบัติในการแสดงพิกเซลได้แตกต่างกันคือตั้งแต่ขนาด 320 x 200 พิกเซล ถึง 1024 x 768 พิกเซล ดังแสดงในตารางที่ 1.1

ตารางที่ 1.1 ไดรเวอร์ โหมด รีโซลูชัน จำนวนสี และเพจของระบบกราฟิกส์ต่าง ๆ

ไดรเวอร์	โหมด	รีโซลูชัน	สี	เพจ
CGA	CGAC0	320 x 200	C0	1
	CGAC1	320 x 200	C1	1
	CGAC2	320 x 200	C2	1
	CGAC3	320 x 200	C3	1
	CGAHi	640 x 200	2	1
MCGA	MCGAC0	320 x 200	C0	1
	MCGAC1	320 x 200	C1	1
	MCGAC2	320 x 200	C2	1
	MCGAC3	320 x 200	C3	1
	MCGAMed	640 x 200	2	1
	MCGAHi	640 x 480	2	1
EGA	EGALo	640 x 200	16	4
	EGAHi	640 x 350	16	2
	EGA64Lo	640 x 200	16	1
	EGAA64Hi	640 x 350	4	1
	EGAMonoHi	640 x 350	2	1–2
	VGALo	640 x 200	16	2
	VGAMed	640 x 350	16	2
	VGAHi	640 x 480	16	1
HercMono	HercMonoHi	720 x 348	2	2
ATT400	ATT400C0	320 x 200	C0	1
	ATT400C1	320 x 200	C1	1
	ATT400C2	320 x 200	C2	1

ตารางที่ 1.1 (ต่อ) ไดรเวอร์ โหมด รีโซลูชัน จำนวนสี และเพจของระบบกราฟิกส์ต่าง ๆ

ไดรเวอร์	โหมด	รีโซลูชัน	สี	เพจ
	ATT400C3	320 x 200	C3	1
	ATT400Med	640 x 200	2	1
	ATT400Hi	640 x 400	2	1
PC3270	PC3270Hi	720 x 350	2	1
8514	IBM8514Lo	640 x 480	256	1
	IBM8514Hi	1024 x 768	256	1

ระบบกราฟิกส์เหล่านี้สามารถแสดงสีได้ตั้งแต่ 2 สีจนถึง 256 สี สำหรับจอภาพที่แสดงได้ 2 สี จะประกอบด้วยสีพื้นซึ่งเป็นสีมืดและสีสว่าง ซึ่งอาจจะเป็นสีเขียว สีขาว หรือสีเหลืองอำพัน ดังนั้นสีที่เรามองเห็นจากจอภาพชนิดนี้จึงมีเพียงสีเดียวเท่านั้น จึงนิยมเรียกชื่อจอภาพและระบบกราฟิกส์ชนิดนี้ว่า จอภาพโมโนโครม (monochrome) ส่วนจอภาพที่สามารถแสดงได้หลายสี เราเรียกว่า จอภาพสี (color)

ไม่ว่าคอมพิวเตอร์จะมีระบบกราฟิกส์เป็นชนิดใดก็ตาม ถ้าเปิดเครื่องด้วย DOS คอมพิวเตอร์จะเริ่มการแสดงผลด้วยเท็กซ์โหมดเสมอ การเปลี่ยนโหมดให้เป็นกราฟิกส์จะทำได้ก็โดยการใช้คำสั่งเฉพาะสำหรับระบบกราฟิกส์ชนิดนั้น เทอร์โบปาสคาลได้จัดให้มีโปรแกรมสำหรับตรวจสอบระบบกราฟิกส์ในคอมพิวเตอร์โดยอัตโนมัติ เมื่อทราบชนิดของระบบกราฟิกส์แล้ว เทอร์โบปาสคาลจะเปลี่ยนโหมดให้เป็นกราฟพิกส์โหมดที่ตรงกับระบบนั้นโดยอัตโนมัติเช่นเดียวกัน วิธีการดังกล่าวนี้ผลดี 2 ประการคือ

1. ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้สะดวกและเขียนได้ทันที โดยไม่ต้องทราบล่วงหน้าว่าระบบกราฟิกส์ที่มีอยู่ในคอมพิวเตอร์เครื่องนั้นจะเป็นชนิดใด

2. โปรแกรมซึ่งเขียนขึ้นด้วยระบบกราฟิกส์ชนิดหนึ่ง สามารถนำไปใช้กับระบบกราฟิกส์ชนิดอื่นได้ทันที

ระบบกราฟิกส์ซึ่งเทอร์โบปาสคาลสามารถตรวจสอบได้มีหลายชนิด ดังแสดงในตารางที่ 1.2

ตารางที่ 1.2 ระบบกราฟิกส์ที่ใช้กับเทอร์โบปาสคาลได้

CONST	ไดรเวอร์	ความหมาย	ชื่อไฟล์
1	CGA	Color Graphics Adapter	CGA.BGI
2	MCGA	Multicolor Graphics Array	CGA.BGI
3	EGA	Enhanced Graphics Adapter	EGAVGA.BGI
6	IBM8514	IBM 8514 Graphics	IBM8514.BGI
7	HercMono	Hercules Monochrome Graphics	HERC.BGI
8	ATT400	AT&T 400 Line Graphics	ATT.BGI
9	VGA	Video Graphics Array	EGAVGA.BGI
10	PC3270	IBM PC 3270 Graphics	PC3270.BGI

1.8 กราฟิกส์ไดรเวอร์ (graphics driver)

จากตารางที่ 1.2 ในคอลัมน์ทางขวามือสุดเราจะเห็นไฟล์ประเภท BGI (ย่อมาจาก Borland Graphics Interface) ซึ่งเป็นไฟล์เก็บคำสั่งสำหรับใช้กับระบบกราฟิกส์ที่มีชื่ออยู่ในคอลัมน์ทางซ้ายมือ ไฟล์ชนิดนี้นิยมเรียกว่า *ไดรเวอร์* (driver) หรือตัวกระตุ้น เพราะคำสั่งที่มีอยู่ในไฟล์จะทำหน้าที่กระตุ้นให้มีการแสดงกราฟิกส์ที่จอภาพ เนื่องจากระบบกราฟิกส์มีหลายชนิด แต่ละชนิดมีรายละเอียดการทำงานแตกต่างกัน จึงเป็นเหตุให้ต้องมีไดรเวอร์หลายไฟล์คือ ไฟล์ CGA.BGI ใช้กับกราฟิกส์ชนิด CGA ไฟล์ EGAVGA.BGI ใช้กับกราฟิกส์ชนิด EGA และ VGA เป็นต้น

ในการรันโปรแกรมกราฟิกส์มีเงื่อนไขว่า จะต้องมีไฟล์ไดรเวอร์ BGI ชนิดเดียวกับที่มีอยู่ในคอมพิวเตอร์เครื่องนั้น โดยจัดให้ไฟล์อยู่ในดิสก์หรือในไดเรกทอรีที่โปรแกรมกำลังติดต่อด้วย เพราะโปรแกรมจะต้องใช้คำสั่งจากไฟล์ BGI สำหรับสร้างกราฟิกส์ ถ้าไม่มีไฟล์ที่ต้องการ โปรแกรมจะแจ้งให้ทราบและหยุดทำงาน

1.9 ตัวอักษรในโหมดกราฟิกส์

ตัวอักษรที่ใช้ในเท็กซ์โหมดเก็บอยู่ในหน่วยความจำรอม (ROM) แต่ตัวอักษรในโหมดกราฟิกส์เก็บอยู่ในไฟล์อักษร เทอร์โบปาสคาลได้สร้างตัวอักษรกราฟิกส์ ที่มีความสวยงามและมีลักษณะแตกต่างกันให้เลือกใช้ 4 แบบโดยเก็บไว้ในไฟล์ประเภท CHR รวม 4 ไฟล์คือ TRIP.CHR, LITT.CHR, SANS.CHR และ GOTH.CHR ดังแสดงรายละเอียดในตารางที่ 1.3

ตารางที่ 1.3 รายละเอียดของไฟล์อักษรที่มีในเทอร์โบปาสคาล

คอนสแตนต์	ประเภทอักษร	ชื่อไฟล์
DefaultFont = 0;	บิตแมพ	อยู่ในเทอร์โบปาสคาล
TriplexFont = 1;	สโตรก	TRIP.CHR
SmallFont = 2;	สโตรก	LITT.CHR
SansSerifFont = 3;	สโตรก	SANS.CHR
GothicFont = 4;	สโตรก	GOTH.CHR

1.10 การจัดเตรียมสำหรับแสดงกราฟิกส์

เพื่อช่วยให้การเขียนโปรแกรมแสดงกราฟิกส์เป็นไปได้อย่างสะดวก และป้องกันปัญหาที่อาจจะเกิดขึ้น เราควรจะได้จัดเตรียมและพิจารณาสิ่งต่างๆ ที่เกี่ยวข้องกับการเขียนโปรแกรมกราฟิกส์ดังต่อไปนี้

1.10.1 ไฟล์ที่ต้องใช้ ในการคอมไพล์และ/หรือรันโปรแกรม อาจจะต้องใช้ไฟล์บางไฟล์หรือหลายไฟล์พร้อมกัน เพราะฉะนั้นเพื่อความสะดวกจึงควรจัดเตรียมไฟล์ GRAPH.TPU ไฟล์ชนิด BGI และไฟล์ชนิด CHR ไว้บนดิสก์แผ่นเดียวกันหรือในไดเรกทอรีเดียวกันกับไฟล์ TURBO.EXE และ TURBO.TPL นั่นคือบนดิสก์แผ่นนี้จะประกอบด้วยไฟล์ต่างๆ คือ

TURBO.EXE

TURBO.TPL

GRAPH.TPU

ไฟล์ทุกชื่อที่มีชนิดเป็น BGI

ไฟล์ทุกชื่อที่มีชนิดเป็น CHR

1.10.2 ส่วนประกอบของโปรแกรมกราฟิกส์บนจอภาพ โปรแกรมสำหรับแสดงกราฟิกส์บนจอภาพจะมีส่วนประกอบเพิ่มเติมจากโปรแกรมแสดงเท็กซ์ ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 1.1 แสดงกราฟิกส์เป็นภาพวงกลมอยู่บริเวณกลางจอภาพ ถ้าเราใช้ระบบกราฟิกส์แบบสี เช่น CGA EGA หรือ VGA จะเห็นภาพวงกลมเป็นสีน้ำเงิน แต่ถ้าเป็นระบบกราฟิกส์แบบโมโนโครมจะเห็นเป็นสีตามปกติของจอภาพนั้น

โปรแกรม 1.1

```
PROGRAM FirstGraph;
USES Graph;
VAR  GrDriver,GrMode,GrError  : Integer;
     xCenter, yCenter : Integer;
     Radius           : Word;
BEGIN
  GrDriver := Detect;
  InitGraph(GrDriver,GrMode,'');
  GrError := GraphResult;
  IF GrError <> GrOk
  THEN WriteLn('Graphics error : ', GraphErrorMsg(GrError))
  ELSE
```

```
   BEGIN
      xCenter := GetMaxX DIV 2;
      yCenter := GetMaxY DIV 2;
      Radius  := GetMaxY DIV 4;
      SetColor(Blue);
      Circle(xCenter, yCenter, Radius);
      ReadLn;
      CloseGraph;
   END
END.
```

ส่วนประกอบหลักของโปรแกรมสำหรับการแสดงกราฟิกส์มีดังต่อไปนี้

1. มีการใช้ยูนิต GRAPH โดยการระบุไว้ดังนี้

```
USES GRAPH;
```

2. เริ่มโปรแกรมด้วยการกำหนดให้ทำการตรวจ ชนิดของระบบกราฟิกส์ที่มีอยู่ในเครื่องคอมพิวเตอร์โดยอัตโนมัติด้วยสเตตเมนต์

```
GrDriver  :=  Detect;
```

โดย Detect เป็นคอนสแตนต์ มีค่าเท่ากับ 0 (หมายถึงให้ตรวจอัตโนมัติ) ซึ่งได้มีการกำหนดค่า Detect ไว้ล่วงหน้าในยูนิต GRAPH ดังนั้น GrDriver จึงมีค่าเป็น 0 ด้วย ค่าดังกล่าวนี้ถูกกำหนดให้เป็นพารามิเตอร์ของโพรซีเยอร์ **InitGraph** ซึ่งมีรูปแบบดังนี้

```
InitGraph (VAR GraphDriver : Integer; VAR GraphMode :
                 Integer; PathToDriver : String)
```

โดย GraphDriver เป็นค่าคอนสแตนต์ของระบบกราฟิกส์ ซึ่งกำหนดไว้ในยูนิต GRAPH (ดูที่คอลัมน์ CONST และไดรเวอร์ ในตารางที่ 1.2) ถ้าค่าเป็น 0 หรือ

Detect หมายถึงให้โปรแกรมทำการตรวจชนิดของระบบกราฟิกส์ที่มีอยู่ในเครื่องคอม-พิวเตอร์ และกำหนดค่าให้กับโปรแกรมโดยอัตโนมัติ

GraphMode เป็นค่าคอนสแตนต์ซึ่งเป็นโหมดของระบบกราฟิกส์ มีอยู่ในยูนิต GRAPH (ดูที่คอลัมน์โหมดในตารางที่ 1.1) ถ้า GraphDriver ถูกกำหนดค่าให้เป็น 0 หรือ Detect โปรแกรมจะกำหนดค่า GraphMode โดยอัตโนมัติ

PathToDriver ใช้สำหรับระบุชื่อดิสก์หรือไดเรกทอรีที่มีไฟล์ BGI และ CHR เช่น ถ้าเราเก็บไฟล์ดังกล่าวนี้ไว้ในดิสก์ C ไดเรกทอรีย่อย \TBPAS\GRAPH จะเขียนสเตตเมนต์ได้ดังนี้

```
InitGraph(GrDriver, GrMode,'C:\TBPAS\GRAPH');
```

ถ้า PathToDriver เป็น '' แสดงว่าไฟล์ BGI และ CHR อยู่ในดิสก์และไดเรกทอรีที่กำลังใช้อยู่ขณะนั้น

3. เปลี่ยนโหมดจากเท็กซ์ให้เป็นกราฟิกส์ด้วยโพรซีเยอร์ **InitGraph** ด้วยสเตตเมนต์ต่อไปนี้

```
InitGraph(GrDriver, GrMode, '');
```

โปรแกรมจะทำการตรวจสอบ และเริ่มต้นใช้กราฟิกส์ไดรเวอร์ และโหมดที่เหมาะสมที่สุดโดยอัตโนมัติ

InitGraph จะทำงานหลายอย่างคือนำไดรเวอร์ BGI ที่เหมาะสมเข้ามาในหน่วยความจำ กำหนดค่าต่างๆ ให้กับแวเรียเบิล เริ่มต้นกราฟิกส์โหมด และลบจอภาพพร้อมกันนี้ InitGraph จะกำหนดค่าให้แก่ GrMode ด้วย

4. แสดงกราฟิกส์ต่างๆ ตามต้องการ

5. เมื่อสิ้นสุดการแสดงกราฟิกส์แล้ว จะต้องเปลี่ยนโหมดให้กลับไปเป็นเท็กซ์เหมือนเดิม ด้วยโพรซีเยอร์ **CloseGraph** ซึ่งเขียนเป็นสเตตเมนต์ดังนี้

```
CloseGraph
```

1.10.3 ผลจากการเปลี่ยนโหมดเป็นกราฟิกส์ ในการเปลี่ยนโหมดจากเท็กซ์โหมดเป็นกราฟิกส์โหมดนั้น โพรซีเยอร์ InitGraph จะตรวจสอบผลที่เกิดขึ้นว่าถูกต้องหรือผิดพลาดด้วย เราจะดูผลการเปลี่ยนโหมดได้โดยการใช้ฟังก์ชัน **GraphResult** ค่าของฟังก์ชันเป็นอินติเจอร์ตั้งแต่ 0, –1, –2 ถึง –15 ดังแสดงในตารางที่ 1.4

ตารางที่ 1.4 ผลจากการเปลี่ยนโหมดจากเท็กซ์โหมดเป็นกราฟิกส์โหมด

GraphResult	GraphErrorMsg	ความหมาย
0	No error	ไม่มีความผิดพลาด
–1	(BGI) graphics not installed	ไม่มี BGI
–2	Graphics hardware not detected	ไม่พบระบบกราฟิกส์
–3	Device driver file not found ()	ไม่พบไฟล์ไดรเวอร์
–4	Invalid device driver file ()	ไฟล์ไดรเวอร์ไม่ถูกต้อง
–5	Not enough memory to load driver	หน่วยความจำไม่พอที่จะรับไดรเวอร์
–6	Out of memory in scan fill	หน่วยความจำไม่พอใน scan fill
–7	Out of memory in flood fill	หน่วยความจำไม่พอใน flood fill
–8	Font file not found ()	ไม่พบฟอนต์ไฟล์
–9	Not enough memory to load font	หน่วยความจำไม่พอที่จะรับฟอนต์
–10	Invalid graphics mode for selected driver	เลือกโหมดจากไดรเวอร์ไม่ถูกต้อง
–11	Graphics error	กราฟิกส์ผิดพลาด
–12	Graphics I/O error	I/O ของกราฟิกส์ผิดพลาด
–13	Invalid font file ()	ฟอนต์ไฟล์ไม่ถูกต้อง
–14	Invalid font number	เบอร์ฟอนต์ไม่ถูกต้อง
–15	Invalid device number	เบอร์ไดรเวอร์ไม่ถูกต้อง

มีข้อสังเกตเกี่ยวกับ GraphResult คือเมื่อ GraphResult รับค่าจากโปรแกรมแล้วจะมีการเปลี่ยนค่ากลับไปเป็น 0 ทันที ซึ่งแสดงว่าไม่มีความผิดพลาดเกิดขึ้น

ดังนั้น GraphResult จึงให้ค่าที่ถูกต้องเมื่อเรียกใช้ครั้งแรกเท่านั้น การเรียกใช้ในครั้งต่อมาจะได้ผลเป็น 0 เสมอ ด้วยเหตุนี้เราจึงต้องกำหนดค่าที่ได้จาก GraphResult ให้กับแวเรียเบิลทันทีหลังการเรียกใช้ครั้งแรก ดังตัวอย่างสเตตเมนต์จากโปรแกรม 1.1 คือ

```
GrError := GraphResult;
```

ถ้าค่าของ GrError ไม่เป็น 0 แสดงว่ามีความผิดพลาดเกิดขึ้นตอนเปลี่ยนโหมด เราอาจจะให้โปรแกรมแสดงความผิดพลาดเป็นรหัสหรือเป็นข้อความก็ได้ดังตัวอย่างต่อไปนี้

1. แสดงรหัสผิดพลาดด้วยสเตตเมนต์

```
Write('Graphics error : ',GrError);
```

2. แสดงข้อความผิดพลาดด้วยสเตตเมนต์

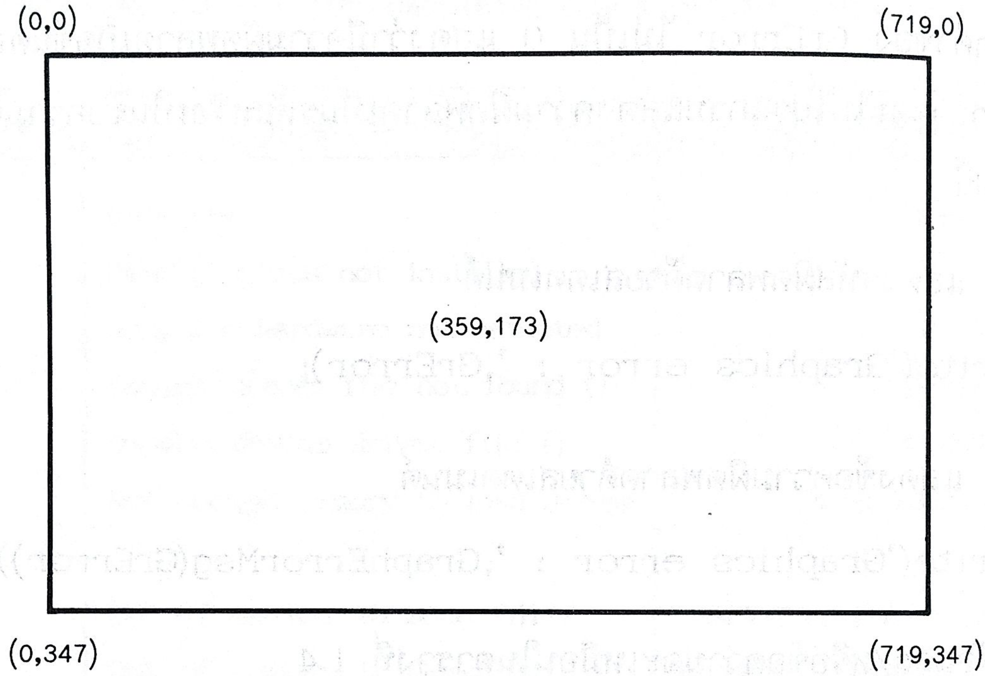
```
Write('Graphics error : ',GraphErrorMsg(GrError));
```

รหัสและ/หรือข้อความจะเหมือนในตารางที่ 1.4

1.11 วิวพอร์ต โคออร์ดิเนต และซีพี

1.11.1 ความหมายของวิวพอร์ต (viewports) โคออร์ดิเนต (coordinates) และซีพี (CP) ภาพแต่ละภาพจะประกอบด้วยพิกเซลจำนวนมากน้อยแตกต่างกัน อย่างไรก็ดีทุกๆ พิกเซลจะมีตำแหน่งที่อยู่ที่แน่นอน ตำแหน่งของพิกเซลเรียกว่าโคออร์ดิเนตค่าของโคออร์ดิเนตจะกำหนดเป็น (X,Y) โดยค่า X เป็นตำแหน่งซึ่งห่างจากทางด้านซ้ายสุดของจอภาพ และ Y เป็นค่าตำแหน่งซึ่งอยู่ห่างจากทางด้านบนสุดของจอภาพ ค่า X และ Y มีหน่วยเป็นพิกเซล ที่ตำแหน่งด้านซ้ายสุดของจอภาพ X มีค่าเป็นศูนย์ X ที่อยู่ทางขวาจะมีค่ามากขึ้นตามลำดับ ทางด้านขวาสุดของจอภาพ X จะมีค่าสูงสุด ที่ตำแหน่งด้านบนสุดของจอภาพ Y มีค่าเป็นศูนย์ Y ที่อยู่ด้านล่างจะมีค่ามาก

ขึ้นตามลำดับ ทางด้านล่างสุดของจอภาพ Y จะมีค่าสูงสุด ค่าสูงสุดของ X+1 และค่าสูงสุดของ Y+1 ก็คือค่ารีโซลูชันของระบบกราฟิกส์ที่มีอยู่ในคอมพิวเตอร์เครื่องนั้นเช่น ระบบเฮอร์คิวลีส (720 x 348) มีค่า X สูงสุดเป็น 719 และค่า Y สูงสุดเป็น 347 ซึ่งหมายความว่า พิกเซลที่อยู่มุมขวาล่างของจอภาพมีโคออร์ดิเนตเป็น (719,347) และพิกเซลที่อยู่มุมซ้ายบนของจอภาพมีโคออร์ดิเนตเป็น (0,0) ดังแสดงในรูปที่ 1.2



รูปที่ 1.2 ค่าโคออร์ดิเนตของระบบกราฟิกส์เฮอร์คิวลิส

บริเวณจอภาพที่สามารถแสดงพิกเซลให้มองเห็นได้ เราเรียกว่า วิวพอร์ต พิกเซลที่อยู่นอกขอบเขตของวิวพอร์ตเราจะมองไม่เห็น ตลอดเวลาที่อยู่ในโหมดกราฟิกส์ เทอร์โบปาสคาลจะเก็บค่าตำแหน่งจุดปัจจุบันหรือซีพี (CP ย่อมาจาก Current Point) ซีพีทำหน้าที่คล้ายเคอร์เซอร์ในเท็กซ์โหมด ความแตกต่างของเคอร์เซอร์กับซีพี ได้แก่ เรามองเห็นเคอร์เซอร์เป็นตัวกะพริบแต่เรามองไม่เห็นซีพี เคอร์เซอร์เลื่อนตำแหน่งได้โดยการกดคีย์และคำสั่ง แต่ซีพีเลื่อนตำแหน่งด้วยคำสั่งเพียงอย่างเดียว หน่วยการเลื่อนเคอร์เซอร์นับเป็นตัวอักษรแต่หน่วยการเลื่อนซีพีนับเป็นพิกเซล

เมื่อเริ่มต้นโหมดกราฟิกส์ ซีพีจะมีค่าเป็น (0,0)

ค่าของซีพีหาได้ด้วยฟังก์ชัน **GetX** และ **GetY**

ค่าสูงสุดของโคออร์ดิเนตที่อยู่ในวิวพอร์ต หาได้ด้วยฟังก์ชัน **GetMaxX** และ **GetMaxY**

การทดสอบว่าซีพีอยู่ในวิวพอร์ตหรือไม่ ทำได้ด้วยสเตตเมนต์ต่อไปนี้

```
IF (0 <= GetX) AND (GetX <= GetMaxX) AND ( 0 <= GetY)
   AND (GetY <= GetMaxY) THEN {ซีพี อยู่ในวิวพอร์ต}
                              ELSE {ซีพี อยู่นอกวิวพอร์ต}
```

การเปลี่ยนตำแหน่งซีพีทำได้โดยใช้โพรซีเยอร์ **MoveTo(X,Y)** ซึ่งจะมีการเลื่อนซีพีจากตำแหน่งปัจจุบันไปอยู่ที่จุด (X,Y) ตามที่ระบุไว้

1.11.2 การลบวิวพอร์ต ทำได้โดยการใช้โพรซีเยอร์ **ClearViewPort** ซึ่งจะมีผลให้จอภาพในบริเวณวิวพอร์ตเปลี่ยนเป็นสีพื้น (background color) ตามสีที่ได้กำหนดไว้ การลบทั้งจอภาพและเลื่อนซีพีไปที่ (0,0) ใช้โพรซีเยอร์ **ClearDe-vice** โพรซีเยอร์ทั้งสองนี้มีการทำงานคล้ายกับโพรซีเยอร์ ClrScr ในเท็กซ์โหมด

1.11.3 การเปลี่ยนวิวพอร์ต ทำด้วยเหตุผล 2 ประการคือ เพื่อกำหนดตำแหน่งแสดงกราฟิกส์ใหม่ และเพื่อกำหนดขอบเขตของกราฟิกส์ให้อยู่ในบริเวณที่ต้องการ โพรซีเยอร์ที่ใช้คือ **SetViewPort** ซึ่งมีรูปแบบดังนี้

```
SetViewPort (X1,Y1,X2,Y2 : Integer; Clip : Boolean)
```

โดย X1,Y1 คือมุมซ้ายบน และ X2,Y2 คือมุมขวาล่างของวิวพอร์ตใหม่ ค่าของ X1,X2,Y1,Y2 จะต้องมากกว่าหรือเท่ากับศูนย์ X2 ต้องมากกว่า X1 และ Y2 ต้องมากกว่า Y1 สำหรับ Clip ถ้ากำหนดเป็น ClipOn หรือ True แสดงว่าภาพจะต้องอยู่ภายในวิวพอร์ตนี้เท่านั้น แต่ถ้ากำหนดเป็น ClipOff หรือ False ภาพจะออกนอกวิวพอร์ตได้

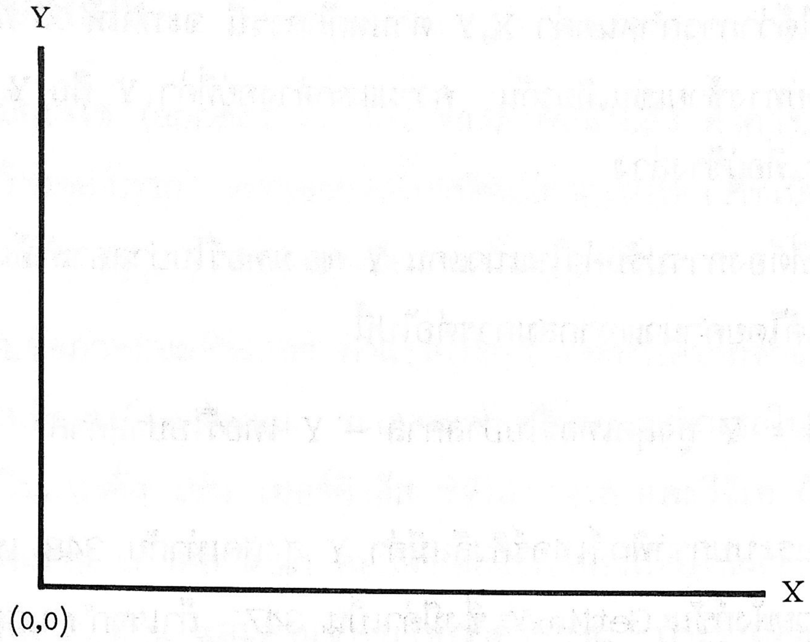
ตัวอย่าง โปรแกรม 1.2 แสดงการกำหนดวิวพอร์ตและ Clip ซึ่งเริ่มต้นด้วยการกำหนดให้เป็น ClipOn ขอให้สังเกตภาพที่เกิดขึ้น เมื่อเปรียบเทียบกับการกำหนด Clip ให้เป็น ClipOff

โปรมแกรม 1.2

```
PROGRAM RestrictViewPort;
USES  Crt, Graph;
VAR GraphDriver, GraphMode : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   SetViewPort(0,0,200,200,ClipOn);
   WHILE NOT Keypressed DO
   BEGIN
      SetColor(1+Random(GetMaxColor));
      Line(Random(GetMaxX),Random(GetMaxY),
           Random(GetMaxX),Random(GetMaxY));
   END;
   CloseGraph
END.
```

1.12 ค่าในแนวแกน Y บนจอภาพ

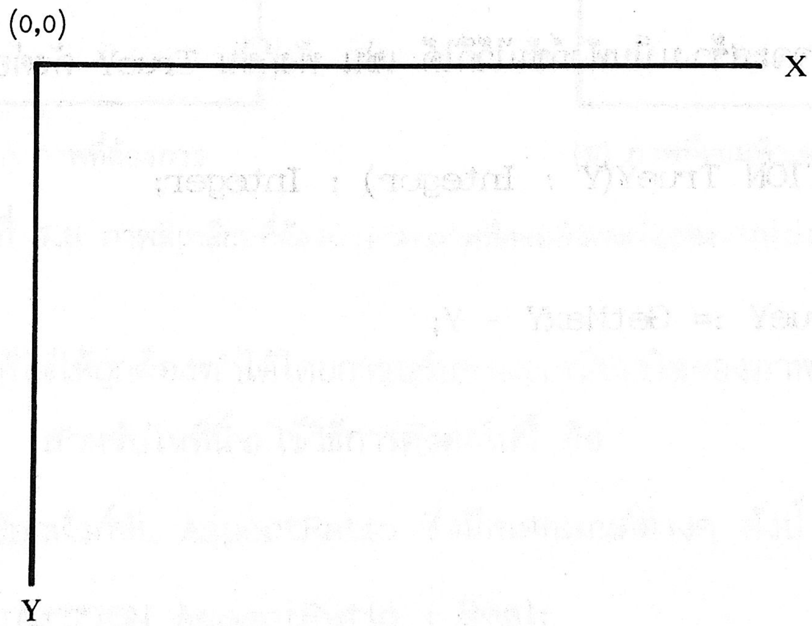
ตามปกติการกำหนดค่าในแนวแกน X และแกน Y มักจะกำหนดให้มุมล่างซ้ายเป็นจุดเริ่มต้น โดยกำหนดให้ X = 0 และ Y = 0 ดังแสดงในรูปที่ 1.3



รูปที่ 1.3 จุดเริ่มต้นของค่าในแนวแกน X และ Y ตามปกติ

การกำหนดค่า X, Y ตามหลักการนี้ จะทำให้ X ที่อยู่ทางขวามีค่ามากกว่า X ที่อยู่ทางซ้าย และ Y ที่อยู่ข้างบนมีค่ามากกว่า Y ที่อยู่ข้างล่าง

สำหรับเทอร์โบปาสคาลได้กำหนดให้มุมบนซ้ายเป็นจุดเริ่มต้น โดยกำหนดให้ X = 0 และ Y = 0 ที่จุดนั้นดังแสดงในรูปที่ 1.4



รูปที่ 1.4 จุดเริ่มต้นของค่าในแนวแกน X และ Y ในเทอร์โบปาสคาล

จะเห็นได้ว่าการกำหนดค่า X,Y ตามหลักการนี้ จะทำให้ X ที่อยู่ทางขวามีค่ามากกว่า X ที่อยู่ทางซ้ายเช่นเดียวกัน ความแตกต่างอยู่ที่ค่า Y คือ Y ที่อยู่ข้างบนจะมีค่าน้อยกว่า Y ที่อยู่ข้างล่าง

ในกรณีที่ต้องการปรับค่าในแนวแกน Y ของเทอร์โบปาสคาลให้เป็นค่า Y ปกติ เราสามารถทำได้โดยคำนวณจากสมการต่อไปนี้

Y ปกติ = Y สูงสุดเทอร์โบปาสคาล – Y เทอร์โบปาสคาล

เช่น ในระบบกราฟิกส์เฮอร์คิวลีสมีค่า Y สูงสุดเท่ากับ 348 เทอร์โบปาสคาลจะอ่านค่านี้ได้ด้วยฟังก์ชัน GetMaxY ซึ่งมีค่าเป็น 347 ถ้าเรากำหนดจุด Y ในโปรแกรมเป็น 200 ค่า Y ปกติจะหาได้ดังนี้

Y ปกติ = 347 – 200

= 147

ในการสร้างภาพด้วยเทอร์โบปาสคาล บางครั้งมีความจำเป็นจะต้องใช้ค่า Y ปกติ ซึ่งเราสามารถหาได้ด้วยสเตตเมนต์ต่อไปนี้

```
Y := GetMaxY - Y
```

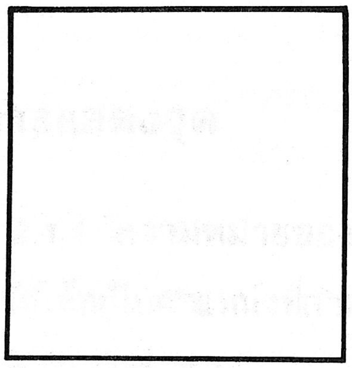
หรืออาจจะสร้างเป็นฟังก์ชันไว้ก็ได้ เช่น ฟังก์ชัน TrueY ดังต่อไปนี้

```
FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;
```

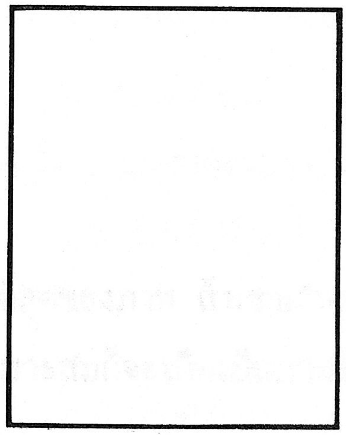
1.13 แอสเป็กเรโช

แอสเป็กเรโช (aspect ratio) ของภาพหมายถึง ค่าความสูงของภาพหารด้วยค่าความกว้างของภาพ ภาพจากจอโทรทัศน์มีค่าแอสเป็กเรโชเป็น 0.75 เนื่องจากจอโทรทัศน์มีความสูง 3 ส่วน และมีความกว้าง 4 ส่วน

สำหรับจอภาพคอมพิวเตอร์ ค่าแอสเป็กเรโชจะหาได้จากจำนวนพิกเซลในแนวตั้ง และจำนวนพิกเซลในแนวนอน ระบบกราฟิกส์ที่มีจำนวนพิกเซลในแนวนอนมากกว่าจำนวนพิกเซลในแนวตั้ง เช่น เฮอร์คิวลีส ซีจีเอ อีจีเอ และวีจีเอ ถ้าเราเขียนโปรแกรมแสดงภาพสามเหลี่ยมด้านเท่า คอมพิวเตอร์จะแสดงเป็นภาพสามเหลี่ยมหน้าจั่วคือมีด้านข้างเท่ากัน 2 ด้าน และด้านข้างยาวกว่าด้านฐาน ถ้าเราเขียนโปรแกรมแสดงภาพสี่เหลี่ยมจัตุรัส คอมพิวเตอร์จะแสดงเป็นภาพสี่เหลี่ยมผืนผ้าที่มีความสูงมากกว่าความกว้าง ดังแสดงในรูปที่ 1.5



(ก) ภาพที่ต้องการ



(ข) ภาพที่คอมพิวเตอร์แสดง

รูปที่ 1.5 ภาพสี่เหลี่ยมที่ต้องการและภาพที่คอมพิวเตอร์แสดงจากโปรแกรม

การแก้ไขให้ถูกต้องทำได้โดยการปรับค่าแอสเป็กเรโชของภาพ ซึ่งสามารถทำได้หลายวิธี สำหรับในที่นี้จะใช้วิธีการดังต่อไปนี้ คือ

1. เรียกฟังก์ชัน AspectRatio ซึ่งมีสเตตเมนต์ต่างๆ ดังนี้

```
FUNCTION AspectRatio : Real;
BEGIN
```

```
    GetAspectRatio(Xasp,Yasp);
    AspectRatio := Xasp/Yasp;
END;
```

2. ปรับค่าแอสเป็กเรโชของ Y โดยนำค่า AspectRatio คูณกับค่า Y ดังสเตตเมนต์ต่อไปนี้

```
Y := AspectRatio * Y
```

3. เนื่องจากค่า Y จะต้องเป็นอินติเจอร์ เราจึงต้องเปลี่ยนค่า Y ให้เป็นอินติเจอร์ด้วยฟังก์ชัน Round ซึ่งสเตตเมนต์ในข้อ 2 จะเปลี่ยนเป็นดังนี้

```
Y := Round(AspectRatio * Y)
```

2
การแสดงกราฟิกส์พื้นฐาน

ในบทนี้เราจะได้ศึกษาเกี่ยวกับการเขียนโปรแกรมเบื้องต้นเพื่อแสดงภาพบนจอภาพ ซึ่งประกอบด้วย ภาพจุด ภาพเส้นตรง ภาพหลายเหลี่ยม และภาพโค้ง ภาพดังกล่าวนี้นับเป็นภาพในระดับพื้นฐานที่จะประกอบกันเข้า เพื่อให้เกิดเป็นภาพที่มีความซับซ้อนและสวยงามในโอกาสต่อไป

2.1 การแสดงจุด

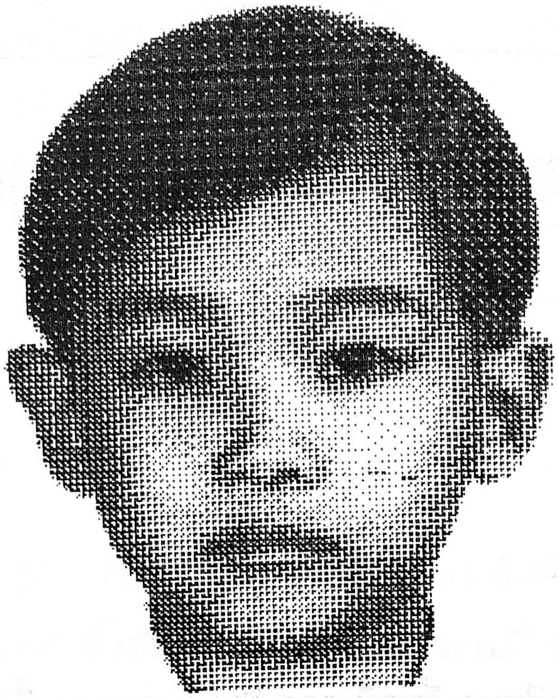
2.1.1 ความหมายของจุด จุดเป็นส่วนเล็กที่สุดของภาพ ถ้าเรานำจุดหลายๆ จุด มาจัดให้อยู่ในตำแหน่งที่กำหนด ด้วยปริมาณที่เหมาะสมก็จะเกิดเป็นภาพให้เห็นได้ดังตัวอย่างในรูปที่ 2.1

2.1.2 วิธีการแสดงจุด การแสดงจุดที่จอภาพของคอมพิวเตอร์เราจะต้องกำหนดค่าให้กับแต่ละจุดอย่างน้อยสองค่าคือ ค่าตำแหน่งที่จุดนั้นอยู่และค่าสีของจุด ไพรซีเยอร์แสดงจุดได้แก่ **PutPixel** ซึ่งมีรูปแบบดังนี้

```
PutPixel (X,Y : Integer; Pixel : Word)
```

โดย X และ Y เป็นค่าตำแหน่งที่จุดนั้นอยู่

Pixel เป็นค่าสีของจุด



รูปที่ 2.1 ภาพที่เกิดจากจุด

2.1.3 ตัวอย่างโปรแกรมกราฟิกส์ที่แสดงด้วยจุด

ตัวอย่าง การแสดงจุดสีน้ำเงินที่ตำแหน่ง 200,100 บนจอภาพ เขียนเป็นโปรแกรมได้ดังโปรแกรม 2.1

โปรแกรม 2.1

```
PROGRAM PixelDot;
USES Graph;
VAR   GraphDriver,GraphMode   : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutPixel(200,100,Blue);
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.2 แสดงจุดที่มีสีต่างๆ ในตำแหน่งต่างๆ บนจอภาพ โดยที่ค่าเหล่านี้ได้มาจากการสุ่ม (random) จากค่าที่เป็นไปได้ของจำนวนจุด และจำนวนสีของระบบกราฟิกส์ที่มีอยู่ในคอมพิวเตอร์เครื่องนั้น

โปรแกรม 2.2

```
PROGRAM PixelStars;
USES Crt,Graph;
VAR  GraphDriver,GraphMode   : Integer;
     X,Y              : Integer;
BEGIN
   Randomize;
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   REPEAT
      X := Random(GetMaxX);
      Y := Random(GetMaxY);
      PutPixel(X,Y,Random(GetMaxColor+1))
   UNTIL KeyPressed;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.3 แสดงภาพฝนตกสาดเป็นสายที่จอภาพในแนวเอียงจากซ้ายไปขวา ฝนแต่ละสายประกอบด้วยจุดจำนวนแตกต่างกัน ซึ่งเป็นจำนวนที่ได้จากค่าสุ่ม สายฝนจะปรากฏในที่ต่างๆ ตามตำแหน่งที่ได้จากค่าสุ่มเช่นเดียวกัน

โปรแกรม 2.3

```
PROGRAM PixelRain;
USES Crt,Graph;
VAR  GraphDriver,GraphMode     : Integer;
     X,Y,PixNo                 : Integer;
BEGIN
   Randomize;
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   REPEAT
      X := Random(GetMaxX);
      Y := Random(GetMaxY);
      FOR PixNo := 1 TO Random(Y) DO
      PutPixel(X-PixNo,Y-PixNo,Random(GetMaxColor+1));
   UNTIL KeyPressed;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.4 แสดงกราฟไซน์ (sine curve) ของมุมที่มีค่าตั้งแต่ 0 ถึง 360 องศา โดยกำหนดให้แสดงครั้งละหนึ่งจุดต่อเนื่องกัน ทำให้มองเห็นกราฟค่อยๆ เกิดขึ้นทีละจุด จนกระทั่งครบทั้งเส้น

โปรแกรม 2.4

```
PROGRAM PixelSine;
USES Crt,Graph;
VAR  GraphDriver,GraphMode     : Integer;
     Degree,SinDegree,Dot      : Integer;
```

```
  Radian                : Real;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   FOR Dot := 1 TO 250 DO PutPixel(0,Dot+20,White);
   FOR Dot := 1 TO 500 DO PutPixel(Dot,150,White);
   FOR Degree := 0 TO 360 DO
   BEGIN
      Radian     := Degree * Pi / 180;
      SinDegree := 150+(-1)*Round(100*Sin(Radian));
      PutPixel(Degree,SinDegree,Blue);
      Delay(10);
   END;
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.5 แสดงการนำจุดมาเรียงให้เป็นรูปสี่เหลี่ยมผืนผ้าเล็กๆ โดยสมมติให้เป็นก้อนอิฐ ต่อจากนั้นจึงแสดงภาพก้อนอิฐหลายก้อนในลักษณะของการเรียงให้เป็นกำแพง

โปรแกรม 2.5

```
PROGRAM PixWall;
USES Graph;
VAR  GraphDriver,GraphMode    : Integer;
     Block, Row, Color, X, Y : Word;

PROCEDURE Brick(X,Y,Color : Integer);
```

```
VAR Horiz,Vert : Word;
BEGIN
   FOR Horiz := X TO X + 95 DO PutPixel(Horiz,Y,Color);
   FOR Horiz := X TO X + 95 DO PutPixel(Horiz,Y + 30,Color);
   FOR Vert  := Y TO Y + 30  DO PutPixel(X,Vert,Color);
   FOR Vert  := Y TO Y + 30  DO PutPixel(X + 95,Vert,Color);
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   FOR Row   := 0 TO 9 DO  { number of row }
   FOR Block := 0 TO 6 DO  { number of brick per row }
   BEGIN
      IF Odd(Row) THEN        { if row is odd number }
      BEGIN
         X := Block * 103 - 50;
         Color := Magenta;
      END
      ELSE                    { row number is event }
      BEGIN
         X := Block * 103;
         Color := Blue
      END;
      Brick(X, Row * 35, Color);
   END;
   ReadLn;
   CloseGraph
END.
```

2.2 การแสดงเส้นตรง

2.2.1 ความหมายของเส้นตรง เส้นตรงเกิดจากการแสดงจุดหลายจุดต่อเนื่องกันในแนวตรง โดยเริ่มจากจุดเริ่มต้น เรียงตามลำดับไปยังจุดสุดท้าย การแสดงเส้นตรงแต่ละเส้นจึงต้องกำหนดจุดเริ่มต้นและจุดสุดท้าย ไพรซีเยอร์แสดงเส้นตรงได้แก่ Line ซึ่งมีรูปแบบดังนี้

```
Line (X1, Y1, X2, Y2 : Integer)
```

โดย X1 และ Y1 เป็นจุดเริ่มต้น

X2 และ Y2 เป็นจุดสุดท้าย

ไพรซีเยอร์ Line ไม่เลื่อนซีพีไปจากตำแหน่งเดิม

ตัวอย่าง การแสดงเส้นตรงซึ่งมีจุดเริ่มต้นที่จุด 0,0 และมีจุดสุดท้ายที่จุด 300,200 เขียนเป็นโปรแกรมได้ดังโปรแกรม 2.6

โปรแกรม 2.6

```
PROGRAM LineLine;
USES Graph;
VAR  GraphDriver,GraphMode    : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Line(0,0,300,200);
   ReadLn;
   CloseGraph
END.
```

2.2.2 สีของเส้นตรง เส้นตรงจะมีสีตามสีที่ถูกกำหนดไว้ ก่อนการสั่งแสดงเส้นตรง การกำหนดสีใช้โพรซีเยอร์ **SetColor** ซึ่งมีรูปแบบดังนี้

SetColor (Color : Word)

โดย Color เป็นค่าของสี มีค่าตั้งแต่ 0 ถึง 15

ตัวอย่าง การแสดงเส้นตรงจากจุด 0,0 ถึงจุดสูงสุดของ X,Y โดยมีสีต่างๆ สลับกันเมื่อใช้กับจอภาพสี สำหรับจอภาพสีเดียวจะแสดงเส้นแบบกะพริบ เขียนโปรแกรมได้ดังโปรแกรม 2.7

โปรแกรม 2.7

```
PROGRAM LineColor;
USES Crt,Graph;
VAR  GraphDriver,GraphMode   : Integer;
     Color                   : Word;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Randomize;
   REPEAT
      Color:=(Random(GetMaxColor+1));
      SetColor(Color);
      Line(0,0,GetMaxX,GetMaxY);
      Delay(100);
   UNTIL KeyPressed;
   CloseGraph
END.
```

2.2.3 ลักษณะของเส้นตรง เส้นตรงจะมีลักษณะตามที่ถูกกำหนดไว้ก่อนการแสดงเส้นตรงนั้น การกำหนดลักษณะของเส้นทำได้โดยการกำหนด แบบ(style) และความหนา (thickness) ของเส้นนั้น การกำหนดแบบและความหนาของเส้นใช้โพรซีเยอร์ **SetLineStyle** ซึ่งมีรูปแบบดังนี้

```
SetLineStyle (LineStyle : Word; Pattern : Word;
              Thickness : Word)
```

โดย LineStyle ถูกกำหนดให้เป็นคอนสแตนต์ ไว้ดังนี้

```
CONST
    SolidLn      =  0; (เส้นทึบ)
    DottedLn     =  1; (เส้นจุด)
    CenterLn     =  2; (เส้นประสั้น)
    DashedLn     =  3; (เส้นประยาว)
    UserBitLn    =  4; (ผู้เขียนโปรแกรมออกแบบเอง)
```

Pattern เป็นแบบของเส้นซึ่งใช้เฉพาะกรณีที่กำหนด LineStyle ให้เป็น UserBitLn (รายละเอียดเรื่องนี้มีในบทที่ 6)

Thickness เป็นความหนาของเส้น ซึ่งกำหนดให้เป็นคอนสแตนต์ไว้ 2 ขนาดดังนี้

```
CONST
    NormWidth    =  1; (ขนาดปกติ)
    ThickWidth   =  3; (หนา 3 เท่าของ NormWidth)
```

ตัวอย่าง โปรแกรม 2.8 แสดงเส้นตรงลากผ่านจอภาพในแนวนอน โดยมีเส้นตรง 3 แบบ ซึ่งมีความหนาแตกต่างกันตามที่ได้กำหนดไว้แล้ว

โปรแกรม 2.8

```
PROGRAM LineStyl;
USES Crt,Graph;
VAR  GraphDriver,GraphMode                 : Integer;
     Style,Pattern,Thickness,LineNo : Word;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Randomize;
   LineNo := 0;
   FOR Style := 0 TO 3 DO
   BEGIN
      FOR Thickness := 1 TO 3 DO
      BEGIN
         SetLineStyle(Style,Patern,Thickness);
         Line(0,LineNo,GetMaxX,LineNo);
         Inc(LineNo,10);
      END;
      Inc(LineNo,20);
   END;
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.9 แสดงเส้นตรงที่มีแบบ ความหนา และความยาว ต่างๆ กัน เส้นตรงเหล่านี้จะแสดงบนจอภาพ ณ ตำแหน่งต่างๆ ตามค่าที่ได้จากการสุ่ม โปรแกรมจะหยุดทำงานเมื่อกดคีย์ใดคีย์หนึ่ง

โปรแกรม 2.9

```
PROGRAM LineRand;
USES Crt,Graph;
VAR  GraphDriver,GraphMode     : Integer;
     X1,Y1,X2,Y2               : Integer;
     Pattern                   : Word;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Randomize;
   REPEAT
      SetColor(Random(GetMaxColor+1));
      SetLineStyle(Random(4),Pattern,Random(4));
      X1 := Random(GetMaxX);  X2 := Random(GetMaxX);
      Y1 := Random(GetMaxY);  Y2 := Random(GetMaxY);
      Line(X1,Y1,X2,Y2);
      Delay(200);
   UNTIL KeyPressed;
   CloseGraph
END.
```

กรณีที่เราต้องการสร้างแบบของเส้นขึ้นใช้เองจะต้องกำหนด LineStyle ให้เป็น UserBitLn และกำหนด Pattern ให้เป็นค่าที่ต้องการ

Pattern เป็นพารามิเตอร์ขนาด 16 บิต ตำแหน่งบิตที่มีค่าเป็น 1 จะเห็นเป็นจุดสว่าง ส่วนตำแหน่งบิตที่มีค่าเป็น 0 จะไม่มีจุดสว่าง

สมมติให้ Pattern มีค่าเป็น $AA (เลขฐานสิบหก)

$A เขียนเป็นบิตได้ดังนี้ 10101010

ลักษณะที่เห็นบนจอภาพเป็นดังนี้ . . . .

$AA เขียนเป็นบิตได้ดังนี้ 1010101010101010

ลักษณะที่เห็นบนจอภาพเป็นดังนี้ . . . .

กรณีที่กำหนดให้เส้นมีความหนาเป็น ThickWidth

$AA เขียนเป็นบิตได้ดังนี้ 1010101010101010

1010101010101010

1010101010101010

ซึ่งจะทำให้เส้นหนาขึ้นดังนี้ . . . .

. . . .

. . . .

ตัวอย่าง โปรแกรม 2.10 แสดงเส้นตรงลากผ่านจอภาพในแนวนอน โดยกำหนดแบบของเส้นให้มีค่าตั้งแต่ $A9 ถึง $AF และมีความหนาแตกต่างกันตามที่ได้กำหนดไว้แล้ว

โปรแกรม 2.10

```
PROGRAM LineBitStyle;
USES Crt,Graph;
VAR  GraphDriver,GraphMode      : Integer;
     Style,Thickness,LineNo     : Word;
BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  Randomize;
```

```
    LineNo := 0;
    FOR Style := $A9 TO $AF DO
    BEGIN
      FOR Thickness := 1 TO 3 DO
      BEGIN
        SetLineStyle(UserBitLn,Style,Thickness);
        Line(0,LineNo,GetMaxX,LineNo);
        Inc(LineNo,10);
      END;
      Inc(LineNo,20);
    END;
    REPEAT UNTIL KeyPressed;
    CloseGraph
END.
```

2.2.4 การแสดงเส้นตรงต่อกัน การแสดงเส้นตรงหลายเส้นต่อเนื่องกันจากจุดหนึ่งไปยังอีกจุดหนึ่งเช่น จากจุด 0,0 ไปที่จุด 50,50 ต่อไปที่ จุด 100,100 และต่อไปที่จุด 300,70 อาจจะเขียนสเตตเมนต์ด้วยโพรซีเยอร์ Line ได้ดังนี้

```
Line (  0,   0,  50,  50, White);
Line ( 50,  50, 100, 100, White);
Line (100, 100, 300,  70, White);
```

จะเห็นได้ว่า เราใช้จุดสุดท้ายของเส้นตรงเส้นก่อนเป็นจุดเริ่มต้นของเส้นตรงเส้นต่อมาเรียงกันตามลำดับ การแสดงเส้นตรงที่ต่อเนื่องกันเช่นนี้อาจจะทำได้สะดวกกว่า โดยการใช้โพรซีเยอร์ LineTo ซึ่งมีรูปแบบดังนี้

```
LineTo (X, Y : Integer)
```

โดย X,Y คือ ตำแหน่งจุดสุดท้ายของเส้นตรงที่ลากมาจากตำแหน่งที่ชี้พีอยู่ ไพรซีเยอร์ LineTo จะแสดงเส้นตรงและเลื่อนซีพีไปที่จุดสุดท้ายของเส้นตรงเส้นสุดท้ายเสมอ จากสเตตเมนต์ Line ในตอนต้น สามารถใช้สเตตเมนต์ LineTo แทนได้ดังนี้

```
LineTo ( 50,50);
LineTo (100,100);
LineTo (300,70);
```

ตัวอย่าง โปรแกรม 2.11 แสดงการลากเส้นตรงต่อเนื่องกันไปโดยมีทิศทางและความยาวของแต่ละเส้นแตกต่างกัน ซึ่งขึ้นอยู่กับค่าที่ได้จากจำนวนสุ่มของโปรแกรม

โปรแกรม 2.11

```
PROGRAM LineToLine;
USES Crt,Graph;
VAR  GraphDriver,GraphMode    : Integer;
     X,Y                      : Integer;
BEGIN
   Randomize;
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   REPEAT
      X := Random(GetMaxX);
      Y := Random(GetMaxY);
      LineTo(X,Y);
      Delay(500);
   UNTIL KeyPressed;
   CloseGraph
END.
```

2.2.5 การแสดงเส้นตรงที่สัมพันธ์กับซีพี เมื่อใช้ซีพีเป็นหลักจะสามารถลากเส้นตรงจากซีพีไปยังตำแหน่งที่อยู่ห่างจากซีพีในทิศทางต่างๆได้ โดยใช้โพรซีเยอร์ **LineRel** ซึ่งมีรูปแบบดังนี้

```
LineRel (Dx, Dy : Integer)
```

โดย Dx และ Dy คือระยะห่างจากซีพีในแนวแกน X และ Y ตามลำดับ ค่าของ Dx และ Dy จะเป็นบวกหรือลบก็ได้

ตัวอย่าง โปรแกรม 2.12 แสดงเส้นตรงซึ่งพุ่งออกไปรอบจุดจุดหนึ่งซึ่งอยู่บริเวณกลางจอภาพ

โปรแกรม 2.12

```
PROGRAM LineRelative;
USES Crt,Graph;
VAR  GraphDriver,GraphMode  : Integer;
     X1,Y1,LineNo           : Integer;

PROCEDURE Center;
BEGIN
   MoveTo(GetMaxX DIV 2,GetMaxY DIV 2);
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   FOR LineNo := -90 TO 90 DO BEGIN Center;
   LineRel( 200, LineNo*10) END;
```

```
   FOR LineNo := -90 TO 90 DO BEGIN Center;
                   LineRel(-200,-LineNo*10) END;
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

2.2.6 การแสดงเส้นตรงในแนวเอียง สำหรับเส้นตรงที่ต้องการแสดงให้เห็นเป็นแนวเอียงขนาดต่างๆ เมื่อเปรียบเทียบกับแนวนอน สามารถสร้างได้โดยใช้ค่าจากที่กำหนดให้รวม 3 ค่าคือ ตำแหน่งจุดเริ่มต้น ความยาวของเส้นตรง และมุมที่วัดจากแนวนอน ดังตัวอย่าง

สมมติให้จุดเริ่มต้นคือ X1,Y1 และจุดสุดท้ายคือ X2,Y2

เส้นตรงยาวเท่ากับ Length

มุมระหว่างแนวนอนกับเส้นตรงเท่ากับ Angle

จะคำนวณหาค่า X2,Y2 ได้ดังนี้

X2 = X1 + Length * Cos(Angle)

Y2 = Y1 + Length * Sin(Angle)

ถ้า X1 = 10

Y1 = 20

Length = 100

Angle = 30 องศา

ค่า X2,Y2 คำนวณได้ดังนี้

X2 = 10 + (100 * Cos(30))

= 10 + (100 * 0.866)

= 96.6

Y2 = 20 + (100 * Sin(30))

= 20 + (100 * 0.5)

= 70

โปรแกรมจะแสดงเส้นตรงเส้นนี้ด้วยโพรซีเยอร์ Line(10,20,96,70)

ตัวอย่าง โปรแกรม 2.13 แสดงเส้นตรงที่มีความยาวเท่ากับ 150 พิกเซลมีจุดเริ่มต้นที่บริเวณกลางจอภาพ เส้นตรงแต่ละเส้นทำมุมกับแนวนอนในขนาดต่างๆ กัน ตั้งแต่ 0 องศา ถึง 280 องศา

โปรแกรม 2.13

```
PROGRAM LineAngle1;
USES Crt,Graph;
VAR  GraphDriver,GraphMode        : Integer;
     Angle,Length,X1,Y1,X2,Y2     : Integer;

FUNCTION Radian(Degree : Integer) : Real;
BEGIN
   Radian := Degree * Pi / 180;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   X1 := Round(GetMaxX DIV 2);
   Y1 := Round(GetMaxY DIV 2);
   Length := 150;
   FOR Angle := 0 TO 280 DO
   BEGIN
      X2 := X1 + Round(Length * Cos(Radian(Angle)));
      Y2 := Y1 + Round(Length * Sin(Radian(Angle)));
      Line( X1, Y1, X2, Y2);
```

```
   END;
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.14 ปรับปรุงมาจากโปรแกรม 2.13 โดยกำหนดมุมล่างซ้ายของจอภาพให้เป็นตำแหน่ง X = 0 และ Y = 0 ซึ่งมีผลให้ Y ค่ามากอยู่ด้านบนของจอภาพและ Y ค่าน้อยจะอยู่ด้านล่างของจอภาพ ซึ่งทำให้ทิศทางการแสดงเส้นตรงแตกต่างจากโปรแกรมเดิม

โปรแกรม 2.14

```
PROGRAM LineAng2;
USES Crt,Graph;
VAR  GraphDriver,GraphMode       : Integer;
     Angle,Length,X1,Y1,X2,Y2   : Integer;

FUNCTION Radian(Degree : Integer) : Real;
BEGIN
   Radian := Degree * Pi / 180;
END;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

BEGIN
```

```
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  X1 := Round(GetMaxX DIV 2);
  Y1 := Round(GetMaxY DIV 2);
  Length := 150;
  FOR Angle := 0 TO 280 DO
  BEGIN
     X2 := X1 + Round(Length * Cos(Radian(Angle)));
     Y2 := Y1 + Round(Length * Sin(Radian(Angle)));
     Line( X1, Y1, X2, TrueY(Y2));
  END;
  REPEAT UNTIL KeyPressed;
  CloseGraph
END.
```

2.3 การแสดงภาพหลายเหลี่ยม

2.3.1 การแสดงภาพสามเหลี่ยม เราสามารถแสดงภาพสามเหลี่ยมได้หลายวิธีดังนี้

1. ลากเส้นตรง 3 เส้นให้ปลายของเส้นหนึ่งเชื่อมต่อกับปลายของอีกเส้นหนึ่งโดยใช้โพรซีเยอร์ Line ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.15 แสดงภาพสามเหลี่ยมซึ่งประกอบด้วยเส้นตรง 3 เส้นคือ

เส้นที่ 1 X1 = 30, Y1 = 40, X2 = 200, Y2 = 60

เส้นที่ 2 X1 = 200, Y1 = 60, X2 = 100, Y2 = 150

เส้นที่ 3 X1 = 100, Y1 = 150, X2 = 30, Y2 = 40

โปรแกรม 2.15

```
PROGRAM LineTriangle;
USES Graph;
VAR  GraphDriver,GraphMode      : Integer;
     X1,Y1,X2,Y2                : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   X1 :=  30;  Y1 :=  40;
   X2 := 200;  Y2 :=  60;
   Line(X1,TrueY(Y1),X2,TrueY(Y2));
   X1 := 200;  Y1 :=  60;
   X2 := 100;  Y2 := 150;
   Line(X1,TrueY(Y1),X2,TrueY(Y2));
   X1 := 100;  Y1 := 150;
   X2 :=  30;  Y2 :=  40;
   Line(X1,TrueY(Y1),X2,TrueY(Y2));
   ReadLn;
   CloseGraph
END.
```

2. ลากเส้นตรงเชื่อมต่อจุด 3 จุดโดยลากจากจุดหนึ่งไปยังอีกจุดหนึ่งต่อเนื่องกันไปจนครบทุกจุด โพรซีเยอร์ที่ใช้คือ LineTo ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.16 แสดงภาพสามเหลี่ยมซึ่งเกิดจากการลากเส้นตรงเชื่อมจุด 3 จุดคือ X1 = 30, Y1 = 40, X2 = 200, Y2 = 60 และ X3 = 100, Y3 = 150

โปรแกรม 2.16

```
PROGRAM LineToTriangle;
USES Graph;
VAR  GraphDriver,GraphMode      : Integer;
     X1,Y1,X2,Y2,X3,Y3          : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   X1 :=  30;  Y1 :=  40;
   X2 := 200;  Y2 :=  60;
   X3 := 100;  Y2 := 150;
   MoveTo(X1,TrueY(Y1));
   LineTo(X2,TrueY(Y2));
   LineTo(X3,TrueY(Y3));
   LineTo(X1,TrueY(Y1));
   ReadLn;
   CloseGraph
END
```

ตัวอย่าง โปรแกรม 2.17 แสดงภาพสามเหลี่ยมซึ่งได้จากการรับค่า X1,Y1, X2,Y2 และ X3,Y3 จากคีย์บอร์ด เป็นผลให้โปรแกรมสามารถแสดงภาพสามเหลี่ยมได้หลายแบบ ตามข้อมูลที่ผู้ใช้โปรแกรมป้อนให้แก่คอมพิวเตอร์

โปรแกรม 2.17

```
PROGRAM DrawTriangle;
USES Graph;
VAR  GraphDriver,GraphMode      : Integer;
     X1,Y1,X2,Y2,X3,Y3          : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

BEGIN
   WriteLn('Draw a Triangle');
   Write('X1    = '); ReadLn(X1);
   Write('Y1    = '); ReadLn(Y1);
   Write('X2    = '); ReadLn(X2);
   Write('Y2    = '); ReadLn(Y2);
   Write('X3    = '); ReadLn(X3);
   Write('Y3    = '); ReadLn(Y3);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   MoveTo(X1,TrueY(Y1));
   LineTo(X2,TrueY(Y2));
   LineTo(X3,TrueY(Y3));
```

```
    LineTo(X1,TrueY(Y1));
    ReadLn;
    CloseGraph
END.
```

3. ในกรณีที่เราทราบตำแหน่งจุดมุมของภาพสามเหลี่ยมไม่ครบทั้ง 3 จุดอาจจะคำนวณหาตำแหน่งจุดที่เหลือได้จากค่าที่มีอยู่ เช่น ตำแหน่งจุดเริ่มต้น ความยาวของด้านหนึ่ง หรือมุมหนึ่งของภาพสามเหลี่ยม ดังตัวอย่างต่อไปนี้

ก. ภาพสามเหลี่ยมด้านเท่าที่มีด้านฐานขนานกับแกน X ถ้าทราบค่าจุดเริ่มต้นที่มุมซ้ายและความยาวของด้าน จะสามารถคำนวณหาจุดที่เหลืออีก 2 จุด แล้วนำมาเขียนภาพสามเหลี่ยมได้ดังต่อไปนี้

สมมติให้จุดเริ่มต้นคือ X1,Y1
ความยาวของด้านเป็น Base
จะคำนวณหาจุดที่เหลือคือ X2,Y2 และ X3,Y3 ได้ดังนี้

```
X2 = X1 + Base
Y2 = Y1
X3 = X1 + Base * Cos(60)
Y3 = Y1 + Base * Sin(60)
```

จะเห็นได้ว่า จากค่าที่กำหนดให้แต่ละค่าที่ทราบแล้วคือ มุมของภาพสามเหลี่ยมด้านเท่าเป็น 60 องศา เราสามารถคำนวณหาตำแหน่งจุดอีก 2 จุดได้โดยวิธีการทางเรขาคณิตเบื้องต้น เนื่องจากค่าที่ได้จากการคำนวณมีชนิดเป็นเรียล (real) ซึ่งมีค่าทศนิยมรวมอยู่ด้วย แต่ตำแหน่งบนจอภาพกำหนดเป็นจำนวนเต็ม ดังนั้นค่าที่ได้จากการคำนวณจึงต้องทำให้เป็นจำนวนเต็มด้วยฟังก์ชัน Round

ตัวอย่าง โปรแกรม 2.18 แสดงภาพสามเหลี่ยมด้านเท่าที่มีความยาวของด้านและตำแหน่งเริ่มต้นตามที่ผู้ใช้โปรแกรมกำหนด โดย X1,Y1 คือตำแหน่งมุมซ้าย และ Base คือความยาวของด้านฐาน มีหน่วยเป็นพิกเซล

โปรแกรม 2.18

```
PROGRAM EqTriangle1;
USES Graph;
VAR  GraphDriver,GraphMode     : Integer;
    Height,X1,Y1,X2,Y2,X3,Y3 : Integer;
    Base,Angle : Real;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
  TrueY := GetMaxY - Y;
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;

BEGIN
  Angle := 60;
  WriteLn('Draw Equilateral Triangle');
  Write('X1    = '); ReadLn(X1);
  Write('Y1    = '); ReadLn(Y1);
  Write('Base  = '); ReadLn(Base);
  X2 := X1 + Round(Base);
```

```
  Y2 := Y1;
  X3 := X1 + Round(Base * Cos(Radian(Angle)));
  Y3 := Y1 + Round(Base * Sin(Radian(Angle)));
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  MoveTo(X1,TrueY(Y1));
  LineTo(X2,TrueY(Y2));
  LineTo(X3,TrueY(Y3));
  LineTo(X1,TrueY(Y1));
  ReadLn;
  CloseGraph
END.
```

ภาพที่ได้จากโปรแกรม 2.18 มีลักษณะแตกต่างกันตามคุณสมบัติของระบบกราฟิกส์ที่มีในคอมพิวเตอร์ สำหรับระบบกราฟิกส์ที่มีจำนวนพิกเซลในแนวนอนมากกว่าจำนวนพิกเซลในแนวตั้งเช่น เฮอร์คิวลีส ซีจีเอ อีจีเอ และวีจีเอ เราจะเห็นเป็นภาพสามเหลี่ยมหน้าจั่วคือมีด้านข้างเท่ากัน 2 ด้านและด้านข้างยาวกว่าด้านฐาน วิธีแก้ไขให้ถูกต้องทำได้โดยการปรับค่าแอสเป็กเรโช ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.19 แสดงภาพสามเหลี่ยมด้านเท่าที่ได้ปรับค่าแอสเป็กเรโชแล้ว

โปรแกรม 2.19

```
PROGRAM EqTriangle2;
USES Graph;
VAR  GraphDriver,GraphMode      : Integer;
     X1,Y1,X2,Y2,X3,Y3          : Integer;
     Base,Angle,Height          : Real;
```

```
    Xasp,Yasp : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

FUNCTION AspectRatio : Real;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp;
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;

BEGIN
   Angle := 60;
   WriteLn('Draw Equilateral Triangle');
   Write('X1   = '); ReadLn(X1);
   Write('Y1   = '); ReadLn(Y1);
   Write('Base = '); ReadLn(Base);
   X2 := X1 + Round(Base);
   Y2 := Y1;
   X3 := X1 + Round(Base * Cos(Radian(Angle)));
   Height := Base * Sin(Radian(Angle));
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
```

```
  Y3 := Y1 + Round(AspectRatio * Height);
       {available after init graph}
  MoveTo(X1,TrueY(Y1));
  LineTo(X2,TrueY(Y2));
  LineTo(X3,TrueY(Y3));
  LineTo(X1,TrueY(Y1));
  ReadLn;
  CloseGraph
END.
```

ข. ภาพสามเหลี่ยมหน้าจั่ว ที่มีด้านฐานขนานกับแกน X ถ้าเราทราบค่าความยาวของด้านฐานและมุมที่ฐาน จะสามารถคำนวณหาจุดที่เหลืออีก 2 จุดได้ดังตัวอย่างต่อไปนี้

สมมติให้จุดเริ่มต้นคือ X1, Y1

ความยาวของด้านฐานเป็น Base

มุมที่ฐานเป็น Angle

จะคำนวณหา X2,Y2 และ X3,Y3 ได้ดังนี้

```
X2 = X1 + Base
Y2 = Y1
X3 = X1 + (Base/2)
Y3 = Y1 + (Base/2) * Tan(Angle)
```

เนื่องจากไม่มีฟังก์ชันหาค่า Tan ในเทอร์โบปาสคาล แต่เราทราบว่า

```
Tan(Angle) = Sin(Angle)/Cos(Angle)
```

ดังนั้นค่า Y3 ก็คือ

```
Y3 = Y1 + (Base/2) * ((Sin(Angle)/Cos(Angle))
```

ค่าที่ได้จากการคำนวณบางค่าจะมีชนิดเป็นเรียล ดังนั้นในโปรแกรมจึงต้องเปลี่ยนค่านั้นให้เป็นจำนวนเต็มด้วยฟังก์ชัน Round

---

ตัวอย่าง โปรแกรม 2.20 แสดงภาพสามเหลี่ยมหน้าจั่วขนาดต่างๆตามที่ผู้ใช้โปรแกรมกำหนด ค่าที่ต้องป้อนให้แก่โปรแกรมคือ ค่า X1, Y1 ซึ่งเป็นตำแหน่งมุมซ้าย ค่าความยาวของด้านฐาน (Base) และค่ามุม (Angle) ที่มีหน่วยเป็นองศา

โปรแกรม 2.20

```
PROGRAM IsoscelesTriangle;
USES Graph;
VAR  GraphDriver,GraphMode     : Integer;
     X1,Y1,X2,Y2,X3,Y3         : Integer;
     Base,Angle,Height         : Real;  { valid for Sqr }
     Xasp, Yasp                : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;

FUNCTION AspectRatio : Real;
BEGIN
   GetAspectRatio(Xasp,Yasp);
```

```
   AspectRatio := Xasp/Yasp;
 END;

BEGIN
   WriteLn('Draw Isosceles Triangle');
   Write('X1    = '); ReadLn(X1);
   Write('Y1    = '); ReadLn(Y1);
   Write('Base = '); ReadLn(Base);
   Write('Left Angle (degree) = '); ReadLn(Angle);
   X2 := X1 + Round(Base);
   Y2 := Y1;
   X3 := X1 + Round(Base / 2);
   Height := (Base/2) * (Sin(Radian(Angle))/Cos(Radian(Angle)));
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Y3 := Y1 + Round(AspectRatio * Height);
   MoveTo(X1,TrueY(Y1));
   LineTo(X2,TrueY(Y2));
   LineTo(X3,TrueY(Y3));
   LineTo(X1,TrueY(Y1));
   ReadLn;
   CloseGraph
END.
```

ค. ภาพสามเหลี่ยมใดๆ ที่มีด้านฐานขนานกับแกน X ถ้าเราทราบความยาวของด้านฐาน ความยาวของด้านซ้าย และมุมระหว่างด้านทั้งสองนี้ จะสามารถคำนวณหาจุดต่างๆ ได้ดังตัวอย่างต่อไปนี้

สมมติให้จุดเริ่มต้นคือ X1, Y1

ความยาวของด้านฐานเป็น Base

ความยาวของด้านซ้ายเป็น LeftSide

มุมระหว่างด้านทั้งสองเป็น Angle

จะคำนวณหาค่า X2, Y2 และ X3, Y3 ได้ดังนี้

X2 = X1 + Base

Y2 = Y1

X3 = X1 + (LeftSide * Cos(Angle))

Y3 = Y1 + (LeftSide * Sin(Angle))

ตัวอย่าง โปรแกรม 2.21 แสดงภาพสามเหลี่ยมลักษณะต่างๆ ตามข้อมูลที่ป้อนให้แก่โปรแกรมซึ่งประกอบด้วย ตำแหน่งมุมซ้าย (X1,Y1) ความยาวของด้านฐาน (Base) ความยาวของด้านซ้าย (LeftSide) และค่ามุมระหว่างด้านทั้งสองนี้

โปรแกรม 2.21

```
PROGRAM AnyTriangle;
USES Graph;
VAR  GraphDriver,GraphMode        : Integer;
     X1,Y1,X2,Y2,X3,Y3            : Integer;
     Base,LeftSide,Height,Angle  : Real;
     Xasp, Yasp                   : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
  TrueY := GetMaxY - Y;
END;
```

```
FUNCTION Radian(Degree : Real) : Real;
BEGIN
  Radian := Degree * Pi / 180
END;


FUNCTION AspectRatio : Real;
BEGIN
  GetAspectRatio(Xasp,Yasp);
  AspectRatio := Xasp/Yasp;
END;


BEGIN
  WriteLn('Draw Given Triangle');
  Write('X1       = '); ReadLn(X1);
  Write('Y1       = '); ReadLn(Y1);
  Write('Base          = '); ReadLn(Base);
  Write('Left Side = '); ReadLn(LeftSide);
  Write('Angle         = '); ReadLn(Angle);
  X2 := X1 + Round(Base);
  Y2 := Y1;
  X3 := X1 + Round(LeftSide * Cos(Radian(Angle)));
  Height := LeftSide * Sin(Radian(Angle));
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  Y3 := Y1 + Round(AspectRatio * Height);
  MoveTo(X1,TrueY(Y1));
  LineTo(X2,TrueY(Y2));
  LineTo(X3,TrueY(Y3));
  LineTo(X1,TrueY(Y1));
```

```
   ReadLn;
   CloseGraph
END.
```

2.3.2 การแสดงภาพสี่เหลี่ยม เราสามารถแสดงภาพสี่เหลี่ยมได้หลายวิธีดังนี้

1. ลากเส้นตรง 4 เส้น ให้ปลายของเส้นหนึ่งเชื่อมต่อกับปลายของอีกเส้นหนึ่งโดยใช้โพรซีเยอร์ Line ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.22 แสดงภาพสี่เหลี่ยมซึ่งประกอบด้วยเส้นตรง 4 เส้นคือ

	X1	Y1	X2	Y2
เส้นที่ 1	30	40	200	40
เส้นที่ 2	200	40	200	100
เส้นที่ 3	200	100	30	100
เส้นที่ 4	30	100	30	40

โปรแกรม 2.22

```
PROGRAM QuadLine;
USES Graph;
VAR  GraphDriver,GraphMode       : Integer;
     X1,Y1,X2,Y2                 : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
  TrueY := GetMaxY - Y;
END;
```

```
BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  X1 :=  30;  Y1 :=  40;
  X2 := 200;  Y2 :=  40;
  Line(X1,TrueY(Y1),X2,TrueY(Y2));
  X1 := 200;  Y1 :=  40;
  X2 := 200;  Y2 := 100;
  Line(X1,TrueY(Y1),X2,TrueY(Y2));
  X1 := 200;  Y1 := 100;
  X2 :=  30;  Y2 := 100;
  Line(X1,TrueY(Y1),X2,TrueY(Y2));
  X1 :=  30;  Y1 := 100;
  X2 :=  30;  Y2 :=  40;
  Line(X1,TrueY(Y1),X2,TrueY(Y2));
  ReadLn;
  CloseGraph
END.
```

---

2. ลากเส้นตรงเชื่อมต่อจุด 4 จุด โดยลากจากจุดหนึ่งไปยังอีกจุดหนึ่งเรียงกันตามลำดับ โพรซีเยอร์ที่ใช้คือ LineTo ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.23 แสดงภาพสี่เหลี่ยมซึ่งประกอบด้วยเส้นตรงที่ลากเชื่อมจุด 4 จุดคือ X1 = 30, Y1 = 40, X2 = 200, Y2 = 40

X3 = 200, Y3 = 100, X4 = 30, Y4 = 100

โปรแกรม 2.23

```
PROGRAM QuadLineTo;
USES Graph;
VAR  GraphDriver,GraphMode       : Integer;
     X1,Y1,X2,Y2,X3,Y3,X4,Y4    : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   X1 :=  30;  Y1 :=  40;
   X2 := 200;  Y2 :=  40;
   X3 := 200;  Y3 := 100;
   X4 :=  30;  Y4 := 100;
   MoveTo(X1,TrueY(Y1));
   LineTo(X2,TrueY(Y2));
   LineTo(X3,TrueY(Y3));
   LineTo(X4,TrueY(Y4));
   LineTo(X1,TrueY(Y1));
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.24 แสดงภาพสี่เหลี่ยมตามค่า X1,Y1, X2,Y2, X3,Y3, และ X4,Y4 ที่รับมาจากคีย์บอร์ด ด้วยวิธีนี้ทำให้โปรแกรมสามารถสร้างภาพสี่เหลี่ยมได้หลายแบบ

โปรแกรม 2.24

```
PROGRAM QuadDraw;
USES Graph;
VAR  GraphDriver,GraphMode      : Integer;
     X1,Y1,X2,Y2,X3,Y3,X4,Y4   : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

BEGIN
   WriteLn('Draw a Quadrangle');
   Write('X1    = '); ReadLn(X1);
   Write('Y1    = '); ReadLn(Y1);
   Write('X2    = '); ReadLn(X2);
   Write('Y2    = '); ReadLn(Y2);
   Write('X3    = '); ReadLn(X3);
   Write('Y3    = '); ReadLn(Y3);
   Write('X4    = '); ReadLn(X4);
   Write('Y4    = '); ReadLn(Y4);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   MoveTo(X1,TrueY(Y1));
```

```
    LineTo(X2,TrueY(Y2));
    LineTo(X3,TrueY(Y3));
    LineTo(X4,TrueY(Y4));
    LineTo(X1,TrueY(Y1));
    ReadLn;
    CloseGraph
END.
```

---

3. ในกรณีที่เราทราบตำแหน่งจุดมุมของภาพสี่เหลี่ยมไม่ครบทั้ง 4 จุด อาจจะคำนวณหาตำแหน่งจุดที่เหลือได้ โดยใช้ค่าที่มีอยู่และจากลักษณะเฉพาะของภาพสี่เหลี่ยมชนิดนั้น ดังตัวอย่างต่อไปนี้

ก. ภาพสี่เหลี่ยมจัตุรัส (square) มีด้านยาวเท่ากันทั้ง 4 ด้าน มุมทุกมุมเท่ากับ 90 องศา สำหรับภาพสี่เหลี่ยมจัตุรัสที่มีด้านหนึ่งขนานกับแกน X ถ้าทราบตำแหน่งมุมล่างซ้ายและความยาวของด้าน จะสามารถคำนวณหาจุดที่เหลืออีก 3 จุด ได้ดังตัวอย่างต่อไปนี้

สมมติให้ มุมล่างซ้ายคือ จุด X1,Y1
ความยาวของด้านเป็น Side
จะคำนวณหา จุดมุมล่างขวา (X2,Y2)
จุดมุมบนขวา (X3,Y3) และ
จุดมุมบนซ้าย (X4,Y4) ได้ดังนี้

X2 = X1 + Side
Y2 = Y1
X3 = X2
Y3 = Y1 + Side
X4 = X1
Y4 = Y3

ตัวอย่าง โปรแกรม 2.25 แสดงภาพสี่เหลี่ยมจัตุรัสที่มีขนาดต่างๆ และอยู่ ณ ตำแหน่งต่างๆ บนจอภาพตามที่ผู้ใช้โปรแกรมกำหนด ค่าที่ป้อนให้แก่โปรแกรมมี 2 ค่า คือ ตำแหน่งมุมล่างซ้าย และความยาวของด้าน

โปรแกรม 2.25

```
PROGRAM QuadSquare1;
USES Graph;
VAR  GraphDriver,GraphMode     : Integer;
     X1,Y1,X2,Y2,Side          : Integer;


FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;


BEGIN
   WriteLn('Draw a Square');
   Write('X1   = '); ReadLn(X1);
   Write('Y1   = '); ReadLn(Y1);
   Write('Side = '); ReadLn(Side);
   X2 := X1 + Side;
   Y2 := Y1 + Side;
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   MoveTo(X1,TrueY(Y1));
   LineTo(X2,TrueY(Y1));
   LineTo(X2,TrueY(Y2));
   LineTo(X1,TrueY(Y2));
```

```
   LineTo(X1,TrueY(Y1));
   ReadLn;
   CloseGraph
END.
```

สำหรับระบบกราฟิกส์ ที่มีจำนวนพิกเซลในแนวนอนมากกว่าจำนวนพิกเซลในแนวตั้ง จะเห็นภาพจากโปรแกรมเป็นภาพสี่เหลี่ยมผืนผ้าคือ มีความสูงมากกว่าแนวนอน วิธีแก้ไขให้ถูกต้องทำได้โดยการปรับค่าแอสเป็กเรโช ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.26 แสดงภาพสี่เหลี่ยมจัตุรัสที่ได้ปรับค่าแอสเป็กเรโชแล้ว

โปรแกรม 2.26

```
PROGRAM QuadSquare2;
USES Graph;
VAR  GraphDriver,GraphMode      : Integer;
     X1,Y1,X2,Y2,Side           : Integer;
     Xasp,Yasp                  : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

FUNCTION AspectRatio : Real;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp;
END;
```

```
BEGIN
   WriteLn('Draw a Square');
   Write('X1    = '); ReadLn(X1);
   Write('Y1    = '); ReadLn(Y1);
   Write('Side = '); ReadLn(Side);
   X2 := X1 + Side;
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Y2 := Y1 + Round(AspectRatio * Side);
   MoveTo(X1,TrueY(Y1));
   LineTo(X2,TrueY(Y1));
   LineTo(X2,TrueY(Y2));
   LineTo(X1,TrueY(Y2));
   LineTo(X1,TrueY(Y1));
   ReadLn;
   CloseGraph
END.
```

ข. ภาพสี่เหลี่ยมผืนผ้า (rectangle) มีมุมทุกมุมเท่ากับ 90 องศา และด้านที่อยู่ตรงกันข้ามมีความยาวเท่ากัน สำหรับรูปสี่เหลี่ยมผืนผ้าที่มีด้านหนึ่งขนานกับแกน X ถ้าทราบตำแหน่งจุดมุมล่างซ้ายและความยาวของด้านทั้งสอง จะสามารถคำนวณหาจุดที่เหลืออีก 3 จุดได้ ดังตัวอย่างต่อไปนี้

สมมติให้มุมล่างซ้ายคือ จุด X1, Y1

ความยาว (แนวนอน) เป็น Length

ความกว้าง (แนวตั้ง) เป็น Width

จะคำนวณหา จุดมุมล่างขวา (X2,Y2)

จุดมุมบนขวา (X3,Y3) และ

จุดมุมบนซ้าย (X4,Y4) ได้ดังนี้

X2 = X1 + Length

Y2 = Y1

X3 = X2

Y3 = Y1 + Width

X4 = X1

Y4 = Y3

ตัวอย่าง โปรแกรม 2.27 แสดงภาพสี่เหลี่ยมผืนผ้าที่มีขนาดต่างๆ และอยู่ ณ ตำแหน่งต่างๆ ตามที่ผู้ใช้โปรแกรมกำหนด ค่าที่ป้อนให้แก่โปรแกรมมี 3 ค่าคือ ตำแหน่งมุมล่างซ้าย ความยาวและความกว้างของภาพสี่เหลี่ยม

โปรแกรม 2.27

```
PROGRAM QuadRectangle;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     X1,Y1,Length,Width      : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;
```

```
PROCEDURE Rectangle(X1,Y1,Length,Width : Integer);
VAR X2,X3,X4,Y2,Y3,Y4 : Integer;
BEGIN
   X2 := X1 + Length;
   Y2 := Y1;
   X3 := X2;
   Y3 := Y1 + Width;
   X4 := X1;
   Y4 := Y3;
   X3 := X4 + Length;
   MoveTo(X1,TrueY(Y1));
   LineTo(X2,TrueY(Y2));
   LineTo(X3,TrueY(Y3));
   LineTo(X4,TrueY(Y4));
   LineTo(X1,TrueY(Y1));
END;

BEGIN
   WriteLn('Draw a Rectangle');
   Write('X1     = '); ReadLn(X1);
   Write('Y1     = '); ReadLn(Y1);
   Write('Length = '); ReadLn(Length);
   Write('Width  = '); ReadLn(Width);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Rectangle(X1,Y1,Length,Width);
   ReadLn;
   CloseGraph
END.
```

ค. ภาพสี่เหลี่ยมด้านขนาน (parallelogram) มีด้านตรงกันข้ามขนานกัน 2 คู่เช่นเดียวกับสี่เหลี่ยมผืนผ้า แต่มุมภายในไม่เป็นมุมฉาก สำหรับสี่เหลี่ยมด้านขนานที่มีด้านหนึ่งขนานกับแกน X ถ้าทราบตำแหน่งมุมล่างซ้าย ค่าของมุมล่างซ้ายและความยาวของด้านทั้งสอง จะสามารถคำนวณหาจุดที่เหลืออีก 3 จุดได้ ดังตัวอย่างต่อไปนี้

สมมติให้มุมล่างซ้ายคือ จุด X1,Y1
ความยาว (แนวนอน) เป็น Length
ความกว้าง (แนวตั้ง) เป็น Width
มุมล่างซ้ายเป็น AngLe
จะคำนวณหา จุดมุมล่างขวา (X2,Y2)
จุดมุมบนขวา (X3,Y3) และ
จุดมุมบนซ้าย (X4,Y4) ได้ดังนี้

X2 = X1 + Length
Y2 = Y1
X4 = X1 + Width * Cos(Angle)
Y4 = Y1 + Width * Sin(Angle)
X3 = X1 + X4 + Length
Y3 = Y4

ตัวอย่าง โปรแกรม 2.28 แสดงภาพสี่เหลี่ยมด้านขนานที่มีขนาด ลักษณะ และอยู่ ณ ตำแหน่งต่างๆ ตามที่ผู้ใช้โปรแกรมกำหนด ค่าที่ป้อนให้แก่โปรแกรมมี 4 ค่า คือ ตำแหน่งมุมล่างซ้าย ความยาว ความกว้างและค่าของมุมล่างซ้าย

โปรแกรม 2.28

```
PROGRAM QuadParallelogram;
USES Graph;
VAR  GraphDriver,GraphMode       : Integer;
     X1,Y1,Length,Width,Angle : Integer;
```

```
FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;

PROCEDURE Parallelogram(X1,Y1,Length,Width,Angle : Integer);
VAR X2,X3,X4,Y2,Y3,Y4 : Integer;
BEGIN
   X2 := X1 + Length;
   Y2 := Y1;
   Y3 := Y1 + Round(Width * Sin(Radian(Angle)));
   X4 := X1 + Round(Width * Cos(Radian(Angle)));
   Y4 := Y3;
   X3 := X4 + Length;
   MoveTo(X1,TrueY(Y1));
   LineTo(X2,TrueY(Y2));
   LineTo(X3,TrueY(Y3));
   LineTo(X4,TrueY(Y4));
   LineTo(X1,TrueY(Y1));
END;

BEGIN
   WriteLn('Draw a Parallelogram');
   Write('X1     = '); ReadLn(X1);
   Write('Y1     = '); ReadLn(Y1);
```

```
   Write('Length  =  '); ReadLn(Length);
   Write('Width   =  '); ReadLn(Width);
   Write('Angle   =  '); ReadLn(Angle);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Parallelogram(X1,Y1,Length,Width,Angle);
   ReadLn;
   CloseGraph
END.
```

ง. เทอร์โบปาสคาลมีโพรซีเยอร์สำหรับแสดงภาพสี่เหลี่ยมผืนผ้าคือ Rectangle ซึ่งมีรูปแบบดังนี้

```
Rectangle (X1, Y1, X2, Y2 : Integer)
```

โดย X1,Y1 คือตำแหน่งมุมบนซ้าย

X2,Y2 คือตำแหน่งมุมล่างขวาของภาพสี่เหลี่ยม

มีเงื่อนไขว่า 0 <= X1 < X2 <= GetMaxX

และ 0 <= Y1 < Y2 <= GetMaxY

ตัวอย่าง โปรแกรม 2.29 แสดงภาพสี่เหลี่ยมผืนผ้าที่มีขนาด และตำแหน่งตามที่ผู้ใช้โปรแกรมกำหนด ค่าที่ป้อนให้แก่โปรแกรมมี 2 ค่าคือ ตำแหน่งมุมบนซ้ายและมุมล่างขวาของรูปสี่เหลี่ยมผืนผ้า

โปรแกรม 2.29

```
PROGRAM QuadTurboRect;
USES Graph;
VAR  GraphDriver,GraphMode  : Integer;
```

```
    X1,Y1,X2,Y2                  : Integer;
BEGIN
   WriteLn('Draw a Rectangle');
   WriteLn('X1, Y1 is the upper left corner and');
   WriteLn('X2, Y2 is the lower right corner of the rectangle');
   WriteLn('Please enter X1, Y1, X2, Y2 (integer)');
   Write('X1 = '); ReadLn(X1);
   Write('Y1 = '); ReadLn(Y1);
   Write('X2 = '); ReadLn(X2);
   Write('Y2 = '); ReadLn(Y2);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Rectangle(X1, Y1, X2, Y2);
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.30 แสดงภาพร่างของคอมพิวเตอร์ซึ่งได้จากการใช้โพรซีเยอร์ Rectangle

โปรแกรม 2.30

```
PROGRAM QuadComputer;
USES Crt,Graph;
VAR  GraphDriver,GraphMode  : Integer;
     X,Y,XSize,YSize,Scale  : Integer;
BEGIN
  GraphDriver := Detect;
```

```
  InitGraph(GraphDriver,GraphMode,'');
  REPEAT
     X := Random(GetMaxX);
     Y := Random(GetMaxY);
     Scale := Random(80);
     XSize := X + Scale;
     YSize := Y + Round(Scale * 3 / 8 + 6);
     Rectangle(X,Y,XSize,YSize);
     Rectangle(X+2,Y+1,XSize-2,YSize-1);
     Rectangle(X,YSize+2,XSize,YSize+Round(Scale/4));
     Delay(1000);
  UNTIL KeyPressed;
  CloseGraph
END.
```

2.3.3 การแสดงภาพหลายเหลี่ยม ภาพเหลี่ยมที่มีด้านมากกว่า 4 ด้าน นิยมเรียกชื่อรวมๆ กันเป็นภาพหลายเหลี่ยม (polygon) การแสดงภาพหลายเหลี่ยมสามารถทำได้หลายวิธีดังต่อไปนี้

1. ลากเส้นตรงเท่ากับจำนวนด้านของภาพเหลี่ยม โดยให้ปลายของแต่ละเส้นเชื่อมต่อกันตามลำดับ ด้วยวิธีการนี้เราจะต้องทราบตำแหน่งจุดปลายของเส้นตรงเหล่านั้นทั้งหมด แล้วจึงลากเส้นตรงเชื่อมจุดเหล่านั้น จำนวนด้านของภาพเหลี่ยมกับจุดปลายของเส้นตรงจะมีจำนวนเท่ากัน เช่น ภาพ 5 เหลี่ยมจะมี 5 จุด ภาพ 8 เหลี่ยมจะมี 8 จุด

สำหรับโปรแกรมที่ใช้แสดงภาพหลายเหลี่ยมภาพหนึ่งโดยเฉพาะ อาจจะกำหนดค่าจุดต่างๆ ไว้เป็นคอนสแตนต์(CONST) หรือแวเรียเบิล(VAR) ก็ได้ โดยกำหนดเป็น X1,Y1, X2,Y2, X3,Y3 ไปจนถึง X สุดท้าย,Y สุดท้าย สเตตเมนต์ที่ใช้สำหรับแสดงภาพหลายเหลี่ยมอาจแบ่งเป็น 2 กลุ่มคือ

ก. แสดงด้วยโพรซีเยอร์ Line มีสเตตเมนต์ดังนี้

```
Line (X1, Y1, X2, Y2);
Line (X2, Y2, X3, Y3);
.
.
.
Line (X สุดท้าย, Y สุดท้าย, X1, Y1,);
```

ข. แสดงด้วยโพรซีเยอร์ LineTo มีสเตตเมนต์ดังนี้

```
MoveTo (X1, Y1);
LineTo (X2, Y2);
LineTo (X3, Y3);
.
.
.
LineTo (X สุดท้าย, Y สุดท้าย) ;
LineTo (X1, Y1);
```

ขอให้สังเกตว่า เราจะต้องลากเส้นตรงจากจุด X สุดท้าย,Y สุดท้าย ไปยัง X1,Y1 ซึ่งเป็นจุดเริ่มต้น เพื่อให้เป็นเส้นปิดภาพเหลี่ยมนั้น

โปรแกรมสำหรับแสดงภาพหลายเหลี่ยมซึ่งเปลี่ยนแปลงขนาดได้ตามที่ผู้ใช้โปรแกรมกำหนด จะต้องมีแวเรียเบิลมากเพียงพอที่จะแสดงภาพเหลี่ยมได้หลายขนาด เช่น ภาพ 100 เหลี่ยมจะต้องมีแวเรียเบิลเก็บค่าจุด X,Y รวม 100 ตัว ซึ่งแวเรียเบิลเหล่านี้อาจจะถูกใช้ทั้งหมดหรือเพียงบางส่วนก็ได้ แวเรียเบิลเก็บค่าจุด X,Y อาจกำหนดให้เป็นชนิดอาร์เรย์ (array) หรือพอยน์เตอร์ (pointer) ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.31 แสดงภาพหลายเหลี่ยมซึ่งกำหนดให้แวเรียเบิลสำหรับเก็บค่าจุด X,Y มีชนิดเป็นอาร์เรย์ ในที่นี้อาร์เรย์มีขนาด 100 ซึ่งสามารถเก็บค่าจุดได้ไม่เกิน 99 จุด เนื่องจากจุดสุดท้ายจะต้องกำหนดให้เท่ากับค่า X1, Y1

โปรแกรม 2.31

```
PROGRAM PolygonA;
USES Crt,Graph;
TYPE XyArray = ARRAY [1..20] OF Word;
VAR  GraphDriver,GraphMode     : Integer;
     X,Y : XyArray;
     Side,Count : Word;
     Xasp,Yasp : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

FUNCTION AspectRatio : Real;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE SetPolygon(VAR X, Y : XyArray);
BEGIN
   WriteLn('*******   DRAW A POLYGON   *******');
```

```
   Write('Enter number of sides of polygon ');
   ReadLn(Side);
   FOR Count := 1 TO Side DO
   BEGIN
      Write('X',Count,' = '); ReadLn(X[Count]);
      Write('Y',Count,' = '); ReadLn(Y[Count]);
   END;
   X[Count + 1] := X[1];
   Y[Count + 1] := Y[1];
END;

PROCEDURE DisplayPolygon( X, Y : XyArray);
BEGIN
   FOR Count := 1 TO Side DO
     Line(X[Count],TrueY(Round(AspectRatio * Y[Count])),
          X[Count+1],TrueY(Round(AspectRatio * (Y[Count+1]))));
END;

BEGIN
   SetPolygon(X, Y);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DisplayPolygon(X, Y);
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.32 แสดงภาพหลายเหลี่ยมซึ่งกำหนดให้แวเรียเบิลสำหรับเก็บค่าจุด X,Y มีชนิดเป็นพอยน์เตอร์ ซึ่งสามารถเก็บค่าจุดได้ตามขนาดของหน่วยความจำที่มีอยู่ในบริเวณฮีพ (heap) แวเรียเบิลชนิดพอยน์เตอร์จะใช้หน่วยความจำได้อย่างมีประสิทธิภาพสูงกว่าแวเรียเบิลชนิดอาร์เรย์

โปรแกรม 2.32

```
PROGRAM PolygonP;
USES Crt,Graph;
TYPE
   XyPointer = ^XyRecord;
   XyRecord  = RECORD
                    Visible : Boolean;
                    X       : Integer;
                    Y       : Integer;
                    Next    : XyPointer;
                 END;
VAR
   GraphDriver,GraphMode : Integer;
   Xy      : XyPointer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
```

```
  GetAspectRatio(Xasp,Yasp);
  AspectRatio := Xasp/Yasp
END;

PROCEDURE SetPolygon(VAR Xy : XyPointer);
VAR Last,PtrXy   : XyPointer;
    Side,Count   : Word;
    Xc,Yc,X1,Y1  : Integer;
  PROCEDURE SetPolygon(VisibleC : Boolean; Xc,Yc : Integer);
  BEGIN
    New(PtrXy);
    IF Xy <> Nil THEN Last^.Next := PtrXy;
    IF Xy =  Nil THEN Xy := PtrXy;
    WITH PtrXy^ DO
    BEGIN
      Visible  := VisibleC;
      X        := Xc;
      Y        := Yc
    END;
    Last := PtrXy;
  END;
BEGIN
  WriteLn('*******  DRAW A POLYGON  *******');
  Write('Enter number of sides of polygon ');
  ReadLn(Side);
  Xy := Nil;
  FOR Count := 1 TO Side DO
  BEGIN
    Write('X',Count,' = '); ReadLn(Xc);
    Write('Y',Count,' = '); ReadLn(Yc);
```

```
        IF Count = 1 THEN
        BEGIN
          X1 := Xc; Y1 := Yc;
          SetPolygon(False, Xc, Yc)
        END
        ELSE SetPolygon(True,  Xc, Yc);
    END;
    SetPolygon(True, X1, Y1);
    Last^.Next := Nil;
END;

PROCEDURE DisplayPolygon(Xy : XyPointer);
VAR Select   : XyPointer;
     Xc,Yc    : Integer;
BEGIN
    Select := Xy;
    WHILE Select <> Nil DO
    BEGIN
        Xc   := Select^.X;
        Yc   := Round(AspectRatio * Select^.Y);
        Yc   := TrueY(Yc);
        IF Select^.Visible THEN LineTo(Xc,Yc) ELSE MoveTo(Xc,Yc);
        Select := Select^.Next;
    END;
END;

BEGIN
    SetPolygon(Xy);
    GraphDriver := Detect;
    InitGraph(GraphDriver,GraphMode,'');
```

```
   ClearDevice;
   DisplayPolygon(Xy);
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

---

2. เทอร์โบปาสคาลมีโพรซีเยอร์สำหรับแสดงภาพหลายเหลี่ยมคือ **DnawPoly** ซึ่งมีรูปแบบดังนี้

DrawPoly (NumPoints : Word; VAR PolyPoints)

โดย NumPoints คือจำนวนจุด X, Y

PolyPoints คือค่าของ X, Y แต่ละจุด

สำหรับภาพเหลี่ยมที่มีจำนวนจุดเท่ากับ n จุด จะต้องกำหนดค่าต่อจากค่าสุดท้าย (คือ n + 1) ให้เท่ากับค่า X1, Y1 ดังนี้

PolyPoints [n + 1] := PolyPoints [1]

PolyPoints จะเป็นข้อมูลชนิดใดก็ได้

ตัวอย่าง โปรแกรม 2.33 แสดงภาพสามเหลี่ยมโดยใช้โพรซีเยอร์ DrawPoly ซึ่งกำหนดค่าจุด X,Y ต่างๆไว้เป็นคอนสแตนต์ ขอให้สังเกตว่าเราต้องกำหนดรวม 4 ค่า และค่าสุดท้ายจะเท่ากับค่าแรก

โปรแกรม 2.33

```
PROGRAM PolygonD;
USES Crt,Graph;
CONST Xy : ARRAY[1..4] OF PointType = ((X :  10; Y : 200),
                                       (X : 400; Y : 200),
```

```
                                    (X : 200; Y :  20),
                                    (X :  10; Y : 200));
VAR  GraphDriver,GraphMode      : Integer;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DrawPoly(4, Xy);  { Turbo pascal procedure }
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

ตัวอย่าง จากโปรแกรม 2.31 เราสามารถปรับปรุงส่วนแสดงภาพ เพื่อใช้โพรซีเยอร์ DrawPoly ได้ดังโปรแกรม 2.34 ต่อไปนี้

โปรแกรม 2.34

```
PROGRAM PolygonT;
USES Crt,Graph;
TYPE XyRecord = RECORD
                  X, Y : Word;
                END;
     XyArray  = ARRAY [1..20] OF XyRecord;
VAR  GraphDriver,GraphMode     : Integer;
     XyRec         : XyRecord;
     XyCo          : XyArray;
     Side,Count : Word;
     Xasp,Yasp  : Word;
     AspectRatio: Real;
```

```
FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y;
END;

PROCEDURE SetPolygon(VAR XyCo : XyArray);
BEGIN
   GraphDriver := Detect;     { initialize for get aspectratio }
   InitGraph(GraphDriver,GraphMode,'');
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp / Yasp;
   CloseGraph;
   WriteLn('*******  DRAW A POLYGON  *******');
   Write('Enter number of sides of polygon ');
   ReadLn(Side);
   FOR Count := 1 TO Side DO
   BEGIN
      Write('X',Count,' = '); ReadLn(XyRec.X);
      Write('Y',Count,' = '); ReadLn(XyRec.Y);
      XyRec.Y := TrueY(Round(AspectRatio * XyRec.Y));
      XyCo[Count]  :=  XyRec;
   END;
   XyCo[Count + 1] :=  XyCo[1];
END;

BEGIN
   SetPolygon(XyCo);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DrawPoly(Side + 1, XyCo);  { Turbo pascal procedure }
```

```
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

---

3. สำหรับภาพหลายเหลี่ยมที่มีด้านเท่ากันทุกด้าน สามารถแสดงให้เห็นได้โดยการคำนวณหาจุด X,Y ต่างๆ จากค่าที่กำหนดให้คือ จุดกึ่งกลางของรูป ระยะห่างระหว่างจุดกึ่งกลางรูปกับจุด X,Y และจำนวนด้านของรูปนั้น ดังตัวอย่าง

สมมติให้จุดกึ่งกลางรูปเป็น Xc, Yc

ระยะห่างจาก Xc,Yc ถึงจุด X,Y เป็น Radius

จำนวนด้านเท่ากับ n

จะคำนวณหาค่าจุด X1,Y1 ถึง Xn,Yn ได้ดังนี้

```
X = Xc + Radius * Cos(Angle)
Y = Xc + Radius * Sin(Angle)
```

โดย Angle เป็นค่าของมุมที่เพิ่มขึ้นจาก 0 องศา ครั้งละ 360/n องศา

ถ้ากำหนดให้ Segment = 360/n

ค่าของ Angle = Angle + Segment โดย Angle จะต้องเริ่มจากค่าศูนย์

เนื่องจากต้องคำนวณค่า X,Y เป็นจำนวน n ครั้ง ดังนั้นสเตตเมนต์คำนวณค่า X1,Y1 ถึง Xn,Yn จึงเป็นดังนี้

```
FOR Side  =  1 TO N Do
BEGIN
   Angle := Angle + Segment;
   X     := Xc + Radius * Cos(Angle);
   Y     := Yc + Radius * Sin(Angle);
END;
```

ตัวอย่าง โปรแกรม 2.35 สามารถแสดงภาพหลายเหลี่ยมได้หลายแบบ ตั้งแต่ภาพสามเหลี่ยมจนกระทั่งภาพวงกลมทั้งนี้ขึ้นอยู่กับการกำหนดค่าต่างๆ ให้แก่โปรแกรม ในตัวอย่างนี้กำหนดให้แสดงภาพ 8 เหลี่ยม

โปรแกรม 2.35

```
PROGRAM PolygonE;
USES Crt,Graph;
VAR  GraphDriver,GraphMode  : Integer;
     Xasp,Yasp : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;

FUNCTION AspectRatio : Real;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE DrawPolygon(Xc,Yc,Radius,SideNo : Integer);
VAR  XCo,YCo,Side     : Integer;
```

```
    Segment,Angle     : Real;
BEGIN
   Segment    := Radian(360 / SideNo);
   Angle      := 0;
   XCo        := Xc + Radius;
   MoveTo(XCo, TrueY(Yc));
   FOR Side := 1 TO SideNo DO
   BEGIN
      Angle := Angle + Segment;
      XCo   := Xc + Round(Radius * Cos(Angle));
      YCo   := Yc + Round(AspectRatio * Radius * Sin(Angle));
      LineTo(XCo,TrueY(YCo));
   END;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DrawPolygon(300,170,150,8);
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

2.4 การแสดงภาพโค้ง

2.4.1 ความหมายของเส้นโค้ง เส้นโค้งเกิดจากการแสดงเส้นตรงเชื่อมต่อจุดต่างๆ ที่อยู่ในแนวโค้ง จากจุดเริ่มต้นเรียงตามลำดับไปยังจุดสุดท้าย ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.36 แสดงภาพเส้นโค้งที่เกิดจากการลากเส้นตรงเชื่อมจุดต่างๆ ต่อเนื่องกัน

โปรแกรม 2.36

```
PROGRAM CurveLine;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   MoveTo(200,100);
   LineTo(205,95);
   LineTo(210,92);
   LineTo(215,91);
   LineTo(220,90);
   LineTo(225,91);
   LineTo(230,92);
   LineTo(232,93);
   LineTo(234,94);
   LineTo(235,95);
   LineTo(236,96);
   LineTo(255,120);
   LineTo(260,150);
   LineTo(255,160);
   LineTo(250,170);
   LineTo(240,175);
   LineTo(230,180);
   ReadLn;
   CloseGraph
END.
```

ความเรียบของเส้นโค้งที่สร้างด้วยวิธีนี้ จะแตกต่างกันตามความยาวของเส้นตรงแต่ละเส้น เส้นโค้งที่เกิดจากเส้นตรงสั้นๆ จะมีความเรียบมากกว่า เส้นโค้งที่เกิดจากเส้นตรงเส้นยาว

ตัวอย่าง โปรแกรม 2.37 แสดงการลากเส้นตรงเส้นสั้นๆ เชื่อมจุดต่างๆ ต่อเนื่องกันเพื่อให้มองเห็นเป็นภาพลายก้นหอย โปรแกรมต้องการให้ป้อนค่าหลายค่าคือ จุดศูนย์กลางของก้นหอย รัศมีเริ่มต้น อัตราเพิ่มของรัศมี และจำนวนรอบของก้นหอย

โปรแกรม 2.37

```
PROGRAM CurveSpiral;
USES Crt,Graph;
VAR  GraphDriver,GraphMode : Integer;
     Xc,Yc,Radius,Turn     : Integer;
     RadiusInc : Real;
     Xasp,Yasp : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;
```

```
FUNCTION AspectRatio : Real;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE SetSpiral(VAR Xc,Yc,Radius,Turn : Integer;
                    VAR RadiusInc : Real);
BEGIN
   WriteLn('************  DRAW  CURVE  ************');
   Write('Enter Center Point (X, Y)     : X = '); ReadLn(Xc);
   Write('                              : Y = '); ReadLn(Yc);
   Write('Enter Radius of arc           : '); ReadLn(Radius);
   Write('Enter Radius increment (0.0): '); ReadLn(RadiusInc);
   Write('Enter Number of turns         : '); ReadLn(Turn);
END;

PROCEDURE DrawSpiral(Xc,Yc : Integer; Radius : Real;
                     Turn : Integer; RadiusInc : Real);
VAR  XCo,YCo,ArcSize,Side,StAngle,EndAngle,No  : Integer;
     StartPoint,Angle         : Real;
BEGIN
   StAngle    := 0; EndAngle := 360;
   StartPoint := Radian(StAngle);
   XCo        := Xc + Round(Radius * Cos(StartPoint));
   YCo        := Yc + Round(AspectRatio * Radius * Sin
                 (StartPoint));
   MoveTo(XCo, TrueY(YCo));
   ArcSize    := Round(Abs(EndAngle - StAngle));
   Angle      := StAngle;
```

```
   FOR No      := 1 TO Turn DO  { number of turn }
   BEGIN
      FOR Side := 1 TO ArcSize DO
      BEGIN
         Angle := Angle + 1; Radius := Radius + RadiusInc;
         XCo    := Xc + Round(Radius * Cos(Radian(Angle)));
         YCo    := Yc + Round(AspectRatio * Radius * Sin(Radian
                ((Angle))));
         LineTo(XCo,TrueY(YCo));
      END;
      Sound(800); Delay(20); NoSound;
   END;
END;

BEGIN
   SetSpiral(Xc,Yc,Radius,Turn,RadiusInc);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DrawSpiral(Xc,Yc,Radius,Turn,RadiusInc);
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

2.4.2 การแสดงภาพวงกลม วิธีการแสดงภาพวงกลมมีดังนี้

1. ภาพวงกลมสามารถสร้างได้จากภาพหลายเหลี่ยมด้านเท่าโดยการกำหนดด้านให้มีจำนวนมาก ซึ่งจะมีผลให้ความยาวของแต่ละด้านมีขนาดสั้นมาก ทำให้มองเห็นเป็นภาพวงกลม จากโปรแกรม 2.35 ถ้าเรากำหนดจำนวนด้านให้มีจำนวนตั้งแต่ 360 ขึ้นไปโปรแกรมจะแสดงภาพวงกลม

ตัวอย่าง โปรแกรม 2.38 แสดงภาพหลายเหลี่ยมด้านเท่า ซึ่งกำหนดให้มีจำนวนด้านเพิ่มขึ้นเรื่อยๆ จะเห็นว่าภาพค่อยๆเปลี่ยนจากภาพหลายเหลี่ยม ไปจนกระทั่งกลายเป็นภาพวงกลม

โปรแกรม 2.38

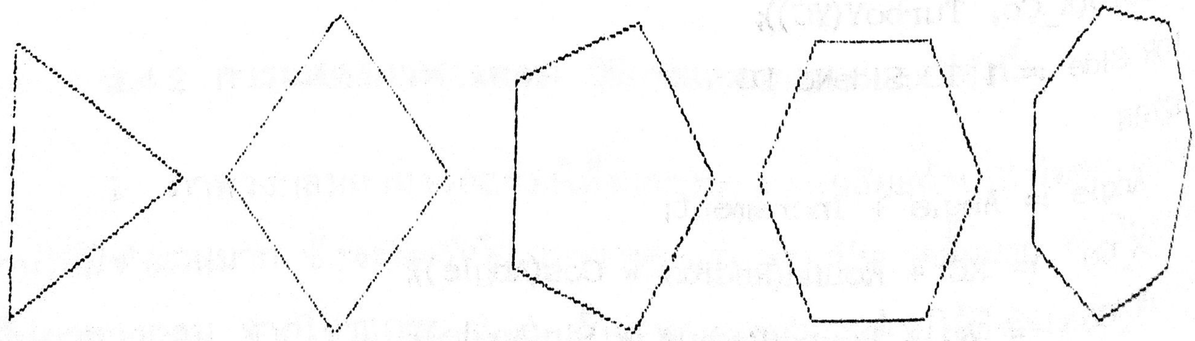
```
PROGRAM MultiPolygon;
USES Graph;
VAR  GraphDriver,GraphMode,PicSide : Integer;

FUNCTION TurboY(Y : Integer) : Integer;
BEGIN
  TurboY := GetMaxY - Y
END;

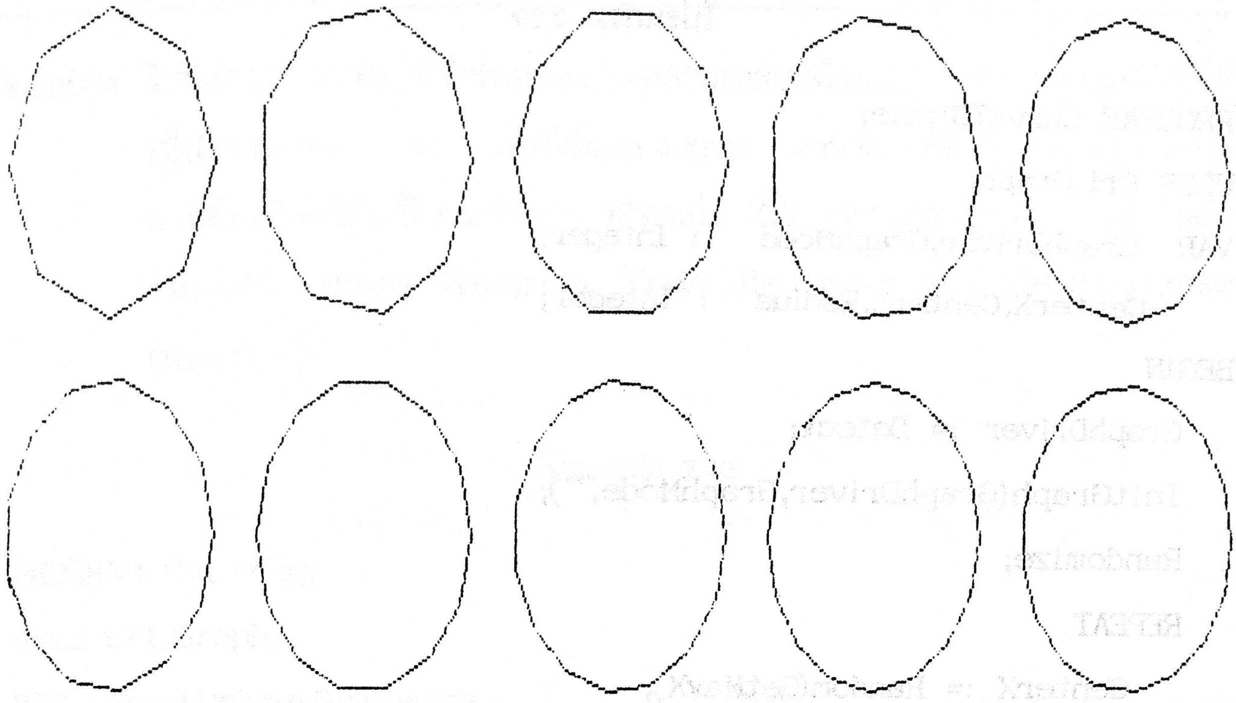
PROCEDURE Polygon(XC,YC,Radius,SideNo : Integer);
CONST Radian = 0.01745;
VAR  X_Co,Y_Co,Side   : Integer;
     Increment,Angle : Real;
BEGIN
  Increment := (360 / SideNo) * Radian;
  Angle := 0;
  X_Co := XC + Radius;
  MoveTo(X_Co, TurboY(YC));
  FOR Side := 1 TO SideNo DO
  BEGIN
    Angle := Angle + Increment;
    X_Co  := XC + Round(Radius * Cos(Angle));
    Y_Co  := YC + Round(Radius * Sin(Angle));
    LineTo(X_Co,TurboY(Y_Co));
```

```
    END;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   FOR PicSide := 3 TO 7 DO
   BEGIN
      Polygon((PicSide-2)*120,TurboY(50),35,PicSide);
   END;
   FOR PicSide := 8 TO 12 DO
   BEGIN
      Polygon((PicSide-7)*120,TurboY(170),35,PicSide);
   END;
   FOR PicSide := 13 TO 17 DO
   BEGIN
      Polygon((PicSide-12)*120,TurboY(290),35,PicSide);
   END;
   ReadLn;
   CloseGraph
END.
```



รูปที่ 2.2 ภาพจากโปรแกรม 2.38



รูปที่ 2.2 (ต่อ) ภาพจากโปรแกรม 2.38

2. เทอร์โบปาสคาลมีโพรซีเยอร์สำหรับแสดงวงกลมคือ **Circle** ซึ่งมีรูปแบบดังนี้

```
Circle (X, Y : Integer;  Radius : Word)
```

โดย X,Y คือตำแหน่งจุดศูนย์กลางของวงกลม

Radius คือความยาวรัศมีของวงกลม

โพรซีเยอร์ Circle ได้ดำเนินการเกี่ยวกับแอสเป็กเรโช ให้ถูกต้องโดยอัตโนมัติตามชนิดของจอภาพที่ใช้

ตัวอย่าง โปรแกรม 2.39 แสดงภาพวงกลมขนาดต่างๆ ที่ตำแหน่งต่างๆ บนจอภาพ โดยการใช้ตัวเลขสุ่มกำหนดค่าของจุดศูนย์กลางและรัศมีของวงกลม

โปรแกรม 2.39

```
PROGRAM CurveTCircle;
USES Crt,Graph;
VAR  GraphDriver,GraphMode     : Integer;
     CenterX,CenterY,Radius    : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Randomize;
   REPEAT
      CenterX := Random(GetMaxX);
      CenterY := Random(GetMaxY);
      Radius  := Random(100);
      Circle(CenterX,CenterY,Radius);
   UNTIL KeyPressed;
   CloseGraph
END.
```

2.4.3 การแสดงอาร์ก (arc) อาร์กคือส่วนหนึ่งของเส้นรอบวงของวงกลมหรือวงรี อาร์กของวงกลมมีวิธีการแสดงดังนี้

1. แสดงด้วยโปรแกรมซึ่งปรับปรุงมาจากโปรแกรมแสดงรูปหลายเหลี่ยมด้านเท่า (โปรแกรม 2.38) รายการที่ปรับปรุงมีดังนี้

ก. กำหนดตำแหน่งเริ่มต้นและตำแหน่งปลายของอาร์กโดยกำหนดค่าเป็นองศาเริ่มจากแกน X ไปในทิศทางทวนเข็มนาฬิกา

ข. กำหนดให้โปรแกรมแสดงเส้นรอบวงของวงกลม ที่มีค่ามุมเพิ่มขึ้นครั้งละ 1 องศา จากค่ามุมเริ่มต้นของอาร์กไปจนถึงมุมปลายของอาร์ก

ตัวอย่าง โปรแกรม 2.38 ซึ่งใช้แสดงภาพหลายเหลี่ยมด้านเท่า สามารถปรับปรุงให้เป็นโปรแกรม 2.40 เพื่อใช้แสดงอาร์ก ตามขนาดที่ผู้ใช้โปรแกรมกำหนด ค่าที่ต้องป้อนให้แก่โปรแกรมคือ ตำแหน่ง X,Y ของจุดศูนย์กลาง มุมเริ่มต้น (Angle) มุมปลายของอาร์ก (EndAngle) และความยาวรัศมีของวงกลม (Radius)

โปรแกรม 2.40

```
PROGRAM CurvePlg;
USES Crt,Graph;
VAR  GraphDriver,GraphMode             : Integer;
     Xc,Yc,Radius,StAngle,EndAngle : Integer;
     Xasp,Yasp : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;

FUNCTION AspectRatio : Real;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;
```

```
PROCEDURE SetPolyCurve(VAR Xc,Yc,StAngle,EndAngle,
                         Radius : Integer);
BEGIN
   WriteLn('************  DRAW ARC   ************');
   Write('Enter Center Point (X, Y)  : X = '); ReadLn(Xc);
   Write('                             : Y = '); ReadLn(Yc);
   Write('Enter Start Angle (Degree) : '); ReadLn(StAngle);
   Write('Enter End   Angle (Degree) : '); ReadLn(EndAngle);
   Write('Enter Radius of arc         : '); ReadLn(Radius);
END;

PROCEDURE DrawPolyCurve(Xc,Yc,StAngle,EndAngle,
                         Radius : Integer);
VAR  XCo,YCo,ArcSize,Side     : Integer;
     StartPoint,Angle    : Real;
BEGIN
   StartPoint := Radian(StAngle);
   XCo         := Xc + Round(Radius * Cos(StartPoint));
   YCo         := Yc + Round(AspectRatio * Radius * Sin
                    (StartPoint));
   MoveTo(XCo, TrueY(YCo));
   ArcSize     := Round(Abs(EndAngle - StAngle));
   Angle       := StAngle;
   FOR Side := 1 TO ArcSize DO
   BEGIN
      Angle := Angle + 1;
      XCo    := Xc + Round(Radius * Cos(Radian(Angle)));
      YCo    := Yc + Round(AspectRatio * Radius * Sin
                  (Radian((Angle))));
      LineTo(XCo,TrueY(YCo));
```

```
   END;
END;

BEGIN
   SetPolyCurve(Xc,Yc,StAngle,EndAngle,Radius);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DrawPolyCurve(Xc,Yc,StAngle,EndAngle,Radius);
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

---

2. เทอร์โบปาสคาลมีโพรซีเยอร์สำหรับแสดงอาร์กคือ **Arc** ซึ่งมีรูปแบบดังนี้

Arc (X, Y : Integer; StAngle, EndAngle, Radius : Word)

โดย X,Y คือตำแหน่งจุดศูนย์กลางของวงกลม

StAngle คือค่ามุมที่จุดเริ่มต้น

EndAngle คือค่ามุมที่ปลายของอาร์ก

Raduis คือรัศมีของวงกลม

ตัวอย่าง โปรแกรม 2.41 แสดงอาร์กของวงกลมสองวง ซึ่งทำให้มองเห็นคล้ายเป็นภาพดวงจันทร์ข้างแรม

โปรแกรม 2.41

```
PROGRAM CurveHalfMoon;
USES Graph;
VAR  GraphDriver,GraphMode     : Integer;
     CenterX,CenterY           : Integer;
```

```
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   CenterX := GetMaxX DIV 2;
   CenterY := GetMaxY DIV 2;
   Arc(CenterX,CenterY,90,270,150);
   Arc(CenterX+200,CenterY,143,217,250);
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.42 แสดงการใช้ภาพวงกลมและอาร์กของวงกลมประกอบกัน เพื่อให้มองเห็นเป็นภาพหน้าการ์ตูนผู้ชาย

โปรแกรม 2.42

```
PROGRAM CurveManFace;
USES Crt,Graph;
VAR  GraphDriver,GraphMode     : Integer;
     CenterX,CenterY           : Integer;

PROCEDURE ManFace(CenterX,CenterY : Word);
BEGIN
   Circle(CenterX,CenterY, 30);
   Arc(CenterX,CenterY,200,350,11);
   Circle(CenterX-8,CenterY-3,2);
   Circle(CenterX+8,CenterY-3,2);
END;
```

```
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   REPEAT
      ManFace(Random(GetMaxX), Random(GetMaxY));
   UNTIL KeyPressed;
   CloseGraph
END.
```

2.4.4 การแสดงภาพวงรี (ellipse) และอาร์กของวงรี ในเทอร์โบปาสคาล ใช้ไพรซีเยอร์ **Ellipse** ซึ่งมีรูปแบบดังนี้

```
Ellipse (X, Y : Integer; StAngle,EndAngle : Word;
         XRadius, YRadius : Word)
```

โดย X,Y คือตำแหน่งจุดกึ่งกลางของวงรี

StAnge คือค่ามุมที่จุดเริ่มต้น

EndAngle คือค่ามุมที่ปลายของอาร์ก

XRadius คือค่ารัศมีในแนวนอน

YRadius คือค่ารัศมีในแนวตั้ง

ถ้ากำหนดให้ StAngle=0 และ EndAngle=360 จะเป็นการแสดงภาพวงรี

ตัวอย่าง โปรแกรม 2.43 แสดงภาพวงรีและอาร์กของวงรีที่มีรูปร่าง ขนาด และตำแหน่งต่างๆ เพื่อให้มองเห็นเป็นภาพฝ่าเท้า

โปรแกรม 2.43

```
PROGRAM CurveElp;
USES Graph;
VAR  GraphDriver,GraphMode      : Integer;
     CenterX,CenterY            : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   CenterX := GetMaxX DIV 2;
   CenterY := GetMaxY DIV 2;
   Ellipse(CenterX,CenterY,150,25,90,100);
   Ellipse(CenterX-70,CenterY-100,0,360,40,50);
   Ellipse(CenterX-11,CenterY-110,0,360,20,35);
   Ellipse(CenterX+30,CenterY-110,0,360,20,30);
   Ellipse(CenterX+65,CenterY-90,0,360,17,28);
   Ellipse(CenterX+90,CenterY-80,0,360,10,25);
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.44 แสดงภาพวงรีและเส้นตรงเพื่อให้มองเห็นเป็นภาพกลุ่มต้นไม้

โปรแกรม 2.44

```
PROGRAM CurveTree;
USES Graph;
VAR  GraphDriver,GraphMode          : Integer;
     XMax,YMax,XRadius,YRadius      : Integer;
```

```
    XCenter,YCenter,Count          : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   XMax      := GetMaxX;
   YMax      := GetMaxY;
   XCenter := XMax DIV 2;
   YCenter := YMax DIV 2 - 20;
   YRadius   := YCenter;
   XRadius   := XCenter DIV 10;
   Ellipse(XCenter,YCenter,0,360,XRadius,YRadius);{midle  tree}

   Ellipse(XCenter + 60, YCenter + 30, 0, 360,     {right trees}
           XRadius + 20, YRadius - 30);
   Ellipse(XCenter + 130, YCenter + 70, 0, 360,
           XRadius + 30, YRadius - 70);
   Ellipse(XCenter + 200, YCenter + 100, 0, 360,
           XRadius + 30, YRadius - 100);

   Ellipse(XCenter - 60, YCenter + 30, 0, 360,    { left trees }
           XRadius + 20, YRadius - 30);
   Ellipse(XCenter - 130, YCenter + 70, 0, 360,
           XRadius + 30, YRadius - 70);
   Ellipse(XCenter - 200, YCenter + 100, 0, 360,
           XRadius + 30, YRadius - 100);

   Line(XCenter - 200, YMax - 40, XCenter - 200, YMax);
   Line(XCenter - 130, YMax - 40, XCenter - 130, YMax);
   Line(XCenter - 60,  YMax - 40, XCenter - 60,  YMax);
   Line(XCenter      , YMax - 40, XCenter      , YMax);
```

```
   Line(XCenter + 200, YMax - 40, XCenter + 200, YMax);
   Line(XCenter + 130, YMax - 40, XCenter + 130, YMax);
   Line(XCenter + 60,  YMax - 40, XCenter + 60,  YMax);
   Line(50, YMax, XMax - 50, YMax);   { ground }
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.45 แสดงภาพวงรี อาร์ก และวงกลม ที่ประกอบกันเข้าเพื่อทำให้มองเห็นเป็นภาพหน้าการ์ตูนผู้หญิง

โปรแกรม 2.45

```
PROGRAM CurveWomanFace;
USES Crt,Graph;
VAR  GraphDriver,GraphMode      : Integer;
     CenterX,CenterY            : Integer;

PROCEDURE Face(CenterX, CenterY : Word);
BEGIN
   Ellipse(CenterX,CenterY,180,0,10,50);
   Arc(CenterX,CenterY+10,220,330,5);
   Circle(CenterX-8,CenterY,2);    {left eye}
   Circle(CenterX+8,CenterY,2);    {right eye}

   Ellipse(CenterX,CenterY+37,90,130,40,50);
   Ellipse(CenterX,CenterY+37,50,90,40,50);
   Ellipse(CenterX,CenterY+37,90,130,35,50);
   Ellipse(CenterX,CenterY+37,50,90,35,50);
```

```
    Ellipse(CenterX,CenterY+37,90,130,30,50);
    Ellipse(CenterX,CenterY+37,50,130,30,50);
    Ellipse(CenterX,CenterY+37,90,130,25,50);
    Ellipse(CenterX,CenterY+37,50,90,25,50);

    Ellipse(CenterX,CenterY+37,90,125,20,50);
    Ellipse(CenterX,CenterY+37,55,90,20,50);
    Ellipse(CenterX,CenterY+37,90,120,15,50);
    Ellipse(CenterX,CenterY+37,60,90,15,50);
END;

BEGIN
    GraphDriver := Detect;
    InitGraph(GraphDriver,GraphMode,'');
    Randomize;
    REPEAT
        Face(Random(GetMaxX), Random(GetMaxY));
    UNTIL KeyPressed;
    ReadLn;
    CloseGraph
END.
```

2.4.5 การแสดงภาพโค้งจากสมการโพลินอเมียล จากโปรแกรมทั้งหมดที่ผ่านมาจะเห็นได้ว่าค่าของ X และ Y เป็นอิสระต่อกัน กล่าวคือการกำหนดค่าหรือการการคำนวณค่าของ X และ Y ไม่ได้เกี่ยวข้องกัน ดังนั้นค่าของ X และ Y จึงไม่มีความสัมพันธ์ต่อกัน สำหรับกรณีที่ค่าของ Y มีความสัมพันธ์กับค่าของ X เช่น ถ้าค่า X มากขึ้น จะทำให้ค่า Y มากขึ้นด้วยนั้น เราสามารถคำนวณหาค่าของ Y ได้จากสมการที่เรียกว่า สมการโพลินอเมียล (polynomial) ซึ่งมีรูปแบบ ดังนี้

$$Y = A_0 + A_1X^1 + A_2X^2 + A_3X^3 + \dots + A_nX^n$$

โดย A_0, A_1 ... A_n เป็นค่าคงตัวเรียกว่าค่าสัมประสิทธิ์

n มีค่าเป็นบวกเรียกว่า ดีกรี (degree) ของโพลินอเมียล

สมการที่แสดงนี้เป็นสมการโพลินอเมียลดีกรี n ค่า X,Y ที่ได้จากสมการทำให้เกิดภาพได้หลายลักษณะทั้งเส้นตรงและเส้นโค้งซึ่งขึ้นอยู่กับดีกรีของสมการ เช่น

ดีกรี 1 สมการคือ

$Y = A_0 + A_1X^1$ เป็นภาพเส้นตรง

ดีกรี 2 สมการคือ

$Y = A_0 + A_1X^1 + A_2X^2$ เป็นภาพพาราโบลา (parabola)

ตัวอย่าง โปรแกรม 2.46 แสดงภาพเส้นตรงจากสมการโพลินอเมียล ดีกรี 1 คือ

$$Y = A_0 + A_1X^1$$

โปรแกรม 2.46

```
PROGRAM PolynomialD1;
USES Graph;
VAR  GraphDriver,GraphMode,Ray : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
  TrueY := GetMaxY - Y
END;

PROCEDURE PolyNmD1(A0,A1 : Real;
                   X0,Y0,StartX,StopX,Segment : Integer);
```

```
VAR  X,Y,IncrementX,Count : Integer;
BEGIN
  IncrementX := (StopX - StartX) DIV Segment;
  X := StartX;
  Y := Round(A0 + A1 * X);
  MoveTo(X + X0, TrueY(Y + Y0));
  FOR Count := 1 TO Segment DO
  BEGIN
    X := X + IncrementX;
    Y := Round(A0 + A1 * X);
    LineTo(X + X0, TrueY(Y + Y0));
  END;
END;

BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  FOR Ray := 1 TO 50 DO
  PolyNmD1(0,Ray/15,GetMaxX DIV 2,0,GetMaxX DIV 8,0,1);
  ReadLn;
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.47 แสดงภาพพาราโบลาจากสมการโพลินอเมียลดีกรี 2 คือ

$$Y = A_0 + A_1X^1 + A_2X^2$$

โปรแกรม 2.47

```
PROGRAM PolyNomialD2;
USES Graph;
VAR  GraphDriver,GraphMode,Ray : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

PROCEDURE PolyNmD2(A0,A1,A2 : Real;
                   X0,Y0,StartX,StopX,Segment : Integer);
VAR  X,Y,IncrementX,Count : Integer;
BEGIN
   IncrementX := (StopX - StartX) DIV Segment;
   X := StartX;
   Y := Round(A0 + A1 * X + A2 * (X * X));
   MoveTo(X + X0, TrueY(Y + Y0));
   FOR Count := 1 TO Segment DO
   BEGIN
      X := X + IncrementX;
      Y := Round(A0 + A1 * X + A2 * ( X * X));
      LineTo(X + X0, TrueY(Y + Y0));
   END;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
```

```
  FOR Ray := 1 TO 100 DO
  PolyNmD2(0,0,0.002*Ray,GetMaxX DIV 2,0,-180,180,20);
  {              width,   base,         length, smoot}
  ReadLn;
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.48 แสดงภาพเส้นโค้งจากสมการไพลินอเมียลดีกรี 3 คือ

$$Y = A_0 + A_1X^1 + A_2X^2 + A_3X^3$$

โปรแกรม 2.48

```
PROGRAM PolyNomialD3;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
  TrueY := GetMaxY - Y
END;

PROCEDURE PolyNmD3(A0,A1,A2,A3 : Real;
                   X0,Y0,StartX,StopX,Segment : Integer);
VAR  X,Y,IncrementX,Count : Integer;
BEGIN
  IncrementX := (StopX - StartX) DIV Segment;
  X := StartX;
  Y := Round(A0 + A1*X + A2*(X*X) + A3*(X*X*X));
```

```
    MoveTo(X + X0, TrueY(Y + Y0));
    FOR Count := 1 TO Segment DO
    BEGIN
       X := X + IncrementX;
       Y := Round(A0 + A1*X + A2*(X*X) + A3*(X*X*X));
       LineTo(X + X0, TrueY(Y + Y0));
    END;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PolyNmD3(0,0,0.02,0.01,GetMaxX DIV 2,GetMaxY DIV 2,-25,25,25);
   ReadLn;
   CloseGraph
END.
```

A_0 เป็นค่าของ Y ณ ตำแหน่งที่เส้นโค้ง(หรือเส้นตรง) ตัดกับแกน Y เรียกว่า ค่าส่วนตัด (intercept) A_1 เป็นค่าความชัน (slope) ของเส้นโค้ง (หรือเส้นตรง) ถ้า A_1 มีค่ามากเส้นจะมีความชันมาก

ภาพโค้งจากสมการโพลินอเมียลเบื้องต้น สามารถปรับปรุงโปรแกรมให้แสดงภาพโค้งซ้ำกันหลายครั้งในลักษณะลูป (loop) เพื่อทำให้เกิดเป็นภาพที่แตกต่างไปจากเดิม เช่น รูปคลื่น รูปเปลือกหอย ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.49 แสดงภาพคลื่นซึ่งเกิดจากการใช้โพรซีเยอร์สร้างเส้นโค้งด้วยสมการโพลินอเมียล ดีกรี 2

โปรแกรม 2.49

```
PROGRAM CurveWave;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
     Xc,Yc,Count,RangeX,StartX,StopX : Integer;
     YIntercept,Slope,A2 : Real;
     Xasp,Yasp : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
  TrueY := GetMaxY - Y
END;

FUNCTION AspectRatio : Real;
BEGIN
  GetAspectRatio(Xasp,Yasp);
  AspectRatio := Xasp/Yasp
END;

PROCEDURE PolyN2(YIntercept,Slope,A2 : Real;
                 Xc,Yc,StartX,StopX,Segment : Integer);
VAR  X,Y,IncrementX,Count : Integer;
BEGIN
  IncrementX := (StopX - StartX) DIV Segment;
  X := StartX;
  Y := Round(AspectRatio * (YIntercept + Slope * X + A2 *
       (X * X)));
  MoveTo(Xc + X, TrueY(Yc + Y));
  FOR Count := 1 TO Segment DO
```

```
    BEGIN
      X := X + IncrementX;
      Y := Round(AspectRatio * (YIntercept + Slope * X + A2 *
           ( X * X)));
      LineTo(Xc + X, TrueY(Yc + Y));
    END;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   YIntercept := 0;  Slope := 0;  A2 := 0.2;
   RangeX := 40;
   StartX := -RangeX DIV 2;
   StopX  :=  RangeX DIV 2;
   Xc := 140;    Yc := 100; { start position of wave }
   FOR Count := 1 TO 8 DO
   BEGIN
      PolyN2(YIntercept,Slope,A2,Xc,Yc,StartX,StopX,20);
      Xc := Xc + RangeX;
   END;
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.50 แสดงภาพโครงสร้างเปลือกหอยเชลล์ ซึ่งเกิดจากการใช้ไพรซีเยอร์สร้างเส้นโค้งด้วยสมการโพลินอเมียล ดีกรี 2

โปรแกรม 2.50

```
PROGRAM PolynmD2Shell;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
     Xc,Yc,Count,StartX,StopX,Increment : Integer;
     A0,A1,A2 : Real;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

PROCEDURE PolyN2(A0,A1,A2 : Real;
                 Xc,Yc,StartX,StopX,Segment : Integer);
VAR  X,Y,IncrementX,Count : Integer;
BEGIN
   IncrementX := (StopX - StartX) DIV Segment;
   X := StartX;
   Y := Round(A0 + A1 * X + A2 * (X * X));
   MoveTo(Xc + X, TrueY(Yc + Y));
   FOR Count := 1 TO Segment DO
   BEGIN
      X := X + IncrementX;
      Y := Round(A0 + A1 * X + A2 * ( X * X));
      LineTo(Xc + X, TrueY(Yc + Y));
   END;
END;

BEGIN
```

```
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  A0 := 0;  A1 := 0;  A2 := 0.02;
  Increment := 30;
  StartX    := -90;
  StopX     := 90;
  Xc := GetMaxX DIV 2; { base point of shell }
  Yc := GetMaxY DIV 4;
  FOR Count := 1 TO 10 DO
  BEGIN
    PolyN2(A0, A1, A2, Xc, Yc, StartX, StopX, 20);
    StartX := StartX - Increment;
    Increment := Increment - 10;
    A2 := A2 * (1 - 1 / (Count + 1));
  END;
  ReadLn;
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 2.51 แสดงภาพโครงสร้างเปลือกหอยเชลล์เหมือนภาพจากโปรแกรม 2.50 แต่เพิ่มเติมให้มีการเปลี่ยนมุมของภาพทุกครั้ง ที่มีการกดคีย์ Enter ซึ่งจะมีภาพทั้งหมดรวม 10 ภาพ

โปรแกรม 2.51

```
PROGRAM CurveShellTransF;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
```

```
    Xc,Yc,Count,StartX,StopX,Increment,PicNo : Integer;
    A0,A1,A2 : Real;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

PROCEDURE PolyN2(A0,A1,A2 : Real;
                 Xc,Yc,StartX,StopX,Segment : Integer);
VAR  X,Y,IncrementX,Count : Integer;
BEGIN
   IncrementX := (StopX - StartX) DIV Segment;
   X := StartX;
   Y := Round(A0 + A1 * X + A2 *(X * X));
   MoveTo(Xc + X, TrueY(Yc + Y));
   FOR Count := 1 TO Segment DO
   BEGIN
      X := X + IncrementX;
      Y := Round(A0 + A1 * X + A2 * ( X * X));
      LineTo(Xc + X, TrueY(Yc + Y));
   END;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   A0 := 0;  A1 := -0.125;  A2 := 0.02;
   FOR PicNo := 1 TO 10 DO
   BEGIN
```

```
    Increment := 30;
    StartX    := -90;
    StopX     := 90;
    Xc := GetMaxX DIV 2; { base point of shell }
    Yc := GetMaxY DIV 4;
    FOR Count := 1 TO 10 DO
    BEGIN
       PolyN2(A0, A1, A2, Xc, Yc, StartX, StopX, 20);
       StartX := StartX - Increment;
       Increment := Increment - 10;
       A2 := A2 * (1 - 1 / (Count + 1));
    END;
    ReadLn;
    ClearViewport;
    A1 := A1+0.125;  A0 := 0;  A2:= 0.02;
  END;{for}
  CloseGraph
END.
```

2.4.6 การแสดงภาพโค้งจากฟังก์ชันตรีโกณมิติ นอกจากจะคำนวณหาค่า Y จากค่าของ X แล้ว เรายังสามารถคำนวณหาค่า Y จากฟังก์ชันตรีโกณมิติ เช่น Sin Cos Tan ได้ด้วย ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.52 แสดงเส้นโค้งแบบไซน์ ซึ่งมีลักษณะเป็นคลื่น โดยกำหนดให้

Height เป็นค่าแอมปลิจูด (amplitude - ความสูงของยอดคลื่น)

Frequency เป็นจำนวนรอบของคลื่นในระยะทาง X

Shift เป็นระยะทางจากแกน X ถึงตำแหน่งเริ่มต้นของคลื่น

(Shift = 0 ถ้าเป็นเส้นโค้งไซน์มาตรฐาน)

โปรแกรม 2.52

```
PROGRAM TrigonSineCurve;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
     Xasp,Yasp : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;

FUNCTION AspectRatio : Real;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE SineCurve(Xc,Yc,StartX,StopX,Segment,
                    Height,Frequency,Shift,YIntercept :
                    Integer);
VAR  X,Y,IncrementX,Count : Integer;
     Angle : Real;
BEGIN
   IncrementX := (StopX - StartX) DIV Segment;
   X := StartX;
```

```
   Angle := Radian(Frequency * X + Shift);
   Y := YIntercept + Round(AspectRatio * (Height *
        Sin(Angle)));
   MoveTo(Xc + X, TrueY(Yc + Y));
   FOR Count := 1 TO Segment DO
   BEGIN
      X := X + IncrementX;
      Angle := Radian(Frequency * X + Shift);
      Y := YIntercept + Round(AspectRatio * (Height *
           Sin(Angle)));
      LineTo(Xc + X, TrueY(Yc + Y));
   END;
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   SineCurve(100,200,0,400,200,70,5,0,0);
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง จากโปรแกรม 2.52 ซึ่งแสดงภาพคลื่นแบบไซน์เบื้องต้น เราสามารถปรับปรุงให้โปรแกรมทำงานเป็นลูป เพื่อให้เกิดภาพคลื่นแบบไซน์ที่มีความสูงแตกต่างกัน ดังโปรแกรม 2.53

โปรแกรม 2.53

```
PROGRAM TrigonSineWave;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
     Xasp,Yasp : Word;
     Y0,Height,CurveNo          : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
   Radian := Degree * Pi / 180
END;

FUNCTION AspectRatio : Real;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE SineCurve(Xc,Yc,StartX,StopX,Segment,
                    Height,Frequency,Shift,YIntercept :
                    Integer);
VAR  X,Y,IncrementX,Count : Integer;
     Angle : Real;
BEGIN
```

```
  IncrementX := (StopX - StartX) DIV Segment;
  X := StartX;
  Angle := Radian(Frequency * X + Shift);
  Y := YIntercept + Round(AspectRatio * (Height *
       Sin(Angle)));
  MoveTo(Xc + X, TrueY(Yc + Y));
  FOR Count := 1 TO Segment DO
  BEGIN
     X := X + IncrementX;
     Angle := Radian(Frequency * X + Shift);
     Y := YIntercept + Round(AspectRatio * (Height *
          Sin(Angle)));
     LineTo(Xc + X, TrueY(Yc + Y));
  END;
END;

BEGIN
  Height := 5;  Y0 := 30;
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  FOR CurveNo := 1 TO 4 DO
  BEGIN
     SineCurve(100,Y0,0,400,200,Height,10,0,0);
     Height := Height * 3;
     Y0 := Y0 + Height;
  END;
  ReadLn;
  CloseGraph
END.
```

2.4.7 การแสดงภาพที่ได้จากค่า x ซึ่งคำนวณจากค่า y ด้วยหลักการเดียวกับการคำนวณหาค่า Y จากค่า X เราสามารถคำนวณหาค่า X จากค่า Y และแสดงเป็นภาพได้ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 2.54 แสดงภาพบันไดเวียน ซึ่งกำหนดให้

FloorHeight เป็นความสูงของห้อง

TreadNo เป็นจำนวนขั้นของบันได

MaxLength เป็นความยาวของบันไดแต่ละขั้น

โปรแกรม 2.54

```
PROGRAM TrigonSineCurve;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
     Xasp,Yasp : Word;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
  TrueY := GetMaxY - Y
END;

FUNCTION Radian(Degree : Real) : Real;
BEGIN
  Radian := Degree * Pi / 180
END;

PROCEDURE SpiralStair(Xc,Yc,FloorHeight,TreadNo,MaxLength,
                      Frequency : Integer);
VAR  X,Y,IncrementY,Count : Integer;
```

```
  Angle : Real;
BEGIN
  IncrementY := FloorHeight DIV TreadNo;
  Y := Yc;
  FOR Count := 1 TO TreadNo DO
  BEGIN
    Angle := Radian(Frequency * Y):
    X := Xc + Round(MaxLength * Sin(Angle));
    MoveTo(Xc, TrueY(Y));
    LineTo(X, TrueY(Y));
    Y := Y + IncrementY;
  END;
END;

BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  SpiralStair(300,1,GetMaxY,86,100,3);
  ReadLn;
  CloseGraph
END.
```

3

การบันทึกภาพและการแสดงภาพในไฟล์

กราฟิกส์และภาพต่างๆ ที่ปรากฏบนจอภาพ เราสามารถบันทึกไว้ในไฟล์ (ในดิสก์) เพื่อใช้ประโยชน์ในโอกาสต่อไปได้ การบันทึกกราฟิกส์ไว้ในไฟล์อาจทำได้ 3 วิธีคือ บันทึกไว้ในโปรแกรม บันทึกไว้ในไฟล์ข้อมูลภาพ และบันทึกไว้ในโปรแกรมและในไฟล์ข้อมูลภาพ ซึ่งมีรายละเอียดดังต่อไปนี้

3.1 การบันทึกภาพไว้ในโปรแกรม

การบันทึกภาพไว้ในโปรแกรม หมายถึงการนำข้อมูลภาพ และวิธีการแสดงภาพมารวมอยู่ในโปรแกรม ซึ่งเมื่อได้จัดการบันทึกโปรแกรมนั้นเอาไว้ในไฟล์แล้ว ก็จะมีไฟล์เพียงไฟล์เดียว การบันทึกกราฟิกส์ด้วยวิธีนี้มีส่วนดีที่สามารถแสดงภาพได้รวดเร็ว เพราะเมื่อสั่งรัน (run) ไปรแกรมนั้น ไปรแกรมจะแสดงภาพทันที ส่วนข้อจำกัดของวิธีการนี้ก็คือ สามารถแสดงภาพได้จำนวนจำกัดเท่าที่มีอยู่ในโปรแกรม การเปลี่ยนแปลงจำนวนภาพ และลักษณะภาพจะทำได้โดยการแก้ไขโปรแกรมเท่านั้น ซึ่งอาจจะไม่สะดวกถ้าเราไม่มีซอร์สโปรแกรม (source program - โปรแกรมต้นฉบับ)

ตามที่เราได้ทราบมาแล้วว่า ภาพกราฟิกส์มีวิธีการสร้าง 2 แบบคือ แบบพิตแมพ และแบบเวกเตอร์ การบันทึกกราฟิกส์แต่ละแบบไว้ในโปรแกรม มีวิธีการดังนี้

3.1.1 การบันทึกภาพแบบบิตแมพไว้ในโปรแกรม สามารถทำได้หลายวิธี แต่วิธีที่สะดวกได้แก่การกำหนดข้อมูลภาพให้เป็นคอนสแตนต์ของโปรแกรม และสร้างโพรซีเยอร์นำข้อมูลภาพมาแสดงเป็นภาพแบบพิตแมพ

ตัวอย่าง โปรแกรม 3.1 แสดงภาพจากข้อมูลภาพที่มีอยู่ในโปรแกรม ข้อมูลภาพมี 52 แถวแต่ละแถวมี 75 ตัวอักษร ตำแหน่งที่เป็นเครื่องหมาย * หรือเครื่องหมายอื่นที่ไม่ใช่เครื่องหมายว่างจะเป็นจุดสว่างบนจอภาพ ส่วนตำแหน่งที่เป็นเครื่องหมายว่างจะไม่เป็นจุดสว่าง โพรซีเยอร์ DisplayPixel ใช้สำหรับอ่านข้อมูลมาแสดงเป็นภาพ โดยมีเงื่อนไขว่า ในข้อมูลแต่ละแถวถ้าไม่เป็นเครื่องหมายว่างให้แสดงจุดสว่าง ด้วยวิธีการนี้เมื่อแสดงได้ครบทุกแถว ก็จะเห็นเป็นภาพ ซึ่งเป็นภาพขนาด 75 x 52 จุด

โปรแกรม 3.1

```
PROGRAM PictureInProgram;
USES Graph;
CONST
  MaxLine      = 52;
  PixelInLine = 75;
  PixelString : ARRAY[1..MaxLine] OF STRING[PixelInLine] = (
```

```
'                              ***********                                         ',
'                           **          *  ***                                     ',
'                        ****     ***     *  **                                    ',
'                    *********    ** **     * **                                   ',
'                  ************    ***       * *                                   ',
'                           ***           *    **                                  ',
'                   **********             * * *                                   ',
'                           **                **                                   ',
'                             ** *           *  * **                               ',
'                              ** *        *  * * ***                              ',
'                            ** *     *    * **** * **                             ',
'                           **         *  * * *** **   ***                         ',
'                          ****  *     *   ***  **      * *                        ',
'**********************  **   * * *   * ***  * *        * **  *********************',
'                         **** * * * * **** * *   *  * * **                        ',
'                         ***** * * ****** ** * *  *   * * *                       ',
'                         ** ************* ** ** *  **** * **                      ',
'                         ** *********** *  ** **  **   ** * *                     ',
'********************* * ********* * * ***    *    ****   * *******************',
'                        ***    **** **     * ** * ****     ** *                   ',
'                          * * *  *   *     * * **  *   *****  **                  ',
'                          ***          *     * *** * ****      * *                ',
'                           * * *            * **  *    ****  *                    ',
'                           ** * *           *  **** ** **     **                  ',
'                            **** * *      *    * **     ***** *                   ',
'                             **** *  * *     *  ***** **     ***                  ',
'                               ****** *     * ** * **** ** **** *                 ',
'                                 ****** **   * *********  **   **                 ',
'                                    ********* ** *********   *** *                ',
'                              * * ***    ********* * * ****  *  *                 ',
'*********************   **  *  ***********   *  *  **************************',
'                          * *** ** **       **** ** ** *****  * *                 ',
'                                               ************* ** * *               ',
'                                                 * ******** * * **  *             ',
'                                                  **** ********   **              ',
'                                                 * ****** * **                    ',
'                                                 ** ** * ** ***                   ',
'                                                  ** **** ** ***                  ',
'                                                 **** * * * * *                   ',
'                                 * *             * *  ** * ** **                  ',
'       *                * *   **  **   * *      *** *    ** * *** * * **          ',
'   *     * * *    **   * *** *     *  ** * *  * *   * * ** * *  *     * * *   * *',
'      *    ** ** *                                *  * * *  * * ** *       **  *  ',
'       * *                 *             * * *    * ** * * * * ***     *      ** **',
'  *       *       * *     *      * *      *  * ** * *   *  **                 *   ',
'        *      * *      *       *           *    * ** * * * ** ***      *         ',
'    *                 *                   *      * ** *   * * ****       *  *     ',
'                *                *      *       *  **** *      * ***   *        *',
'            *            *        *                ***         ****               ',
'                                                       *******                    ',
'                                                                                  ');
```

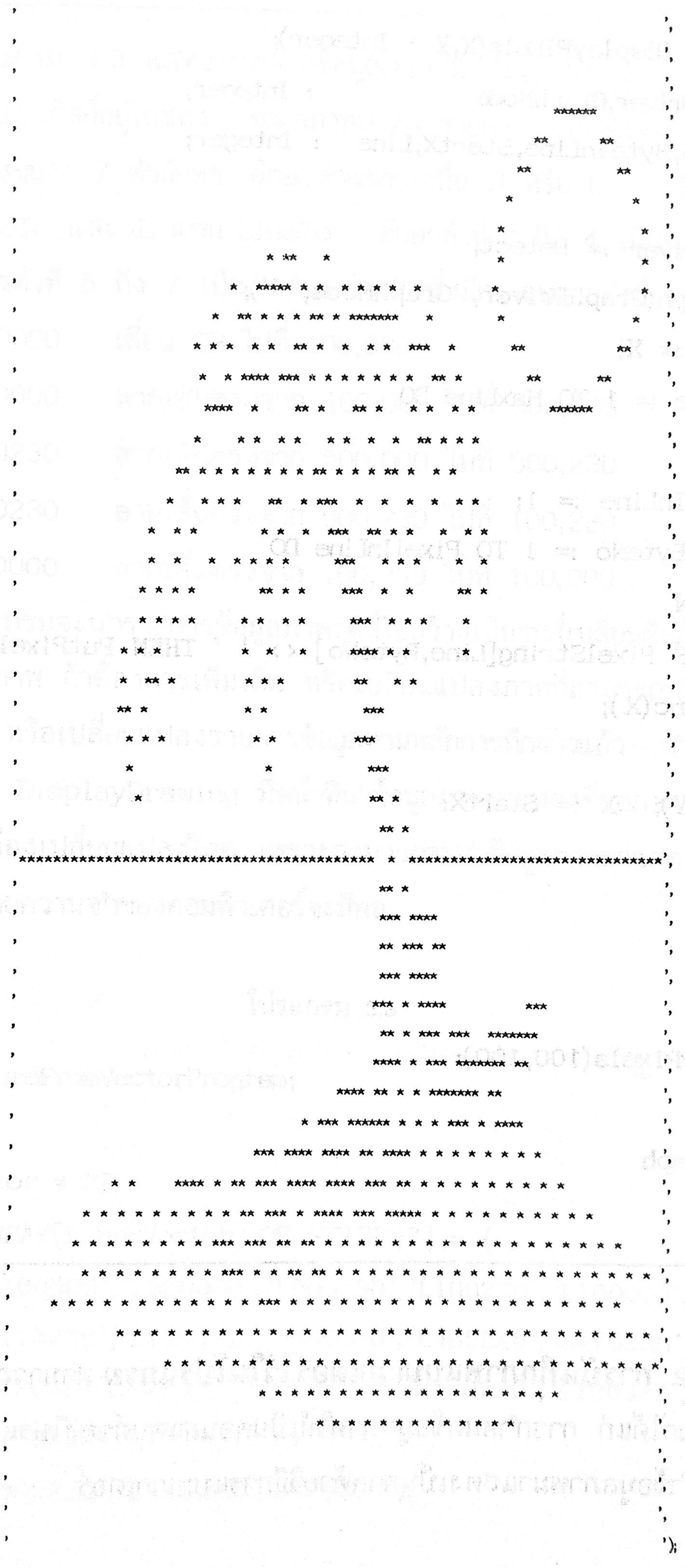
```
    StartX := X;
    FOR Line := 1 TO MaxLine DO
    BEGIN
       ByteInLine := 1;
       FOR ByteNo := 1 TO PixelInLine DO
       BEGIN
          IF PixelString[Line,ByteNo] <> ' ' THEN PutPixel(X,Y,White);
          Inc(X);
       END;
       Inc(Y);   X := StartX;
    END;
END;

BEGIN
   DisplayPixels(100,100);
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 3.2 แสดงภาพจากข้อมูลภาพที่มีอยู่ในโปรแกรม ส่วนประกอบต่างๆ ของโปรแกรมนี้คล้ายกับโปรมแกรม 3.1 มีการเปลี่ยนแปลงเฉพาะข้อมูลภาพให้แตกต่างกัน ภาพในตัวอย่างนี้คือ ภาพเกาะ มีต้นมะพร้าวและดวงจันทร์

โปรแกรม 3.2

```
PROGRAM PictureInProgram2;
USES Graph;
CONST
   MaxLine      = 52;
   PixelInLine  = 75;
   PixelString : ARRAY[1..MaxLine] OF STRING[PixelInLine] = (
```



```
PROCEDURE DisplayPixels(X,Y : Integer);
VAR GraphDriver,GraphMode              : Integer;
    ByteNo,ByteInLine,StartX,Line      : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   StartX := X;
   FOR Line := 1 TO MaxLine DO
   BEGIN
      ByteInLine := 1;
      FOR ByteNo := 1 TO PixelInLine DO
      BEGIN
         IF PixelString[Line,ByteNo] <> ' ' THEN PutPixel(X,Y,White);
         Inc(X);
      END;
      Inc(Y);  X := StartX;
   END;
END;

BEGIN
   DisplayPixels(100,100);
   ReadLn;
   CloseGraph
END.
```

3.1.2 การบันทึกภาพแบบเวกเตอร์ไว้ในโปรแกรม สามารถทำได้หลายวิธี แต่วิธีที่สะดวกได้แก่ การกำหนดข้อมูลภาพให้เป็นคอนสแตนต์ของโปรแกรมและสร้างโพรซีเยอร์นำข้อมูลภาพมาแสดงเป็นภาพด้วยวิธีการแบบเวกเตอร์

ตัวอย่าง โปรแกรม 3.3 แสดงภาพจากข้อมูลภาพที่มีอยู่ในโปรแกรม ภาพที่แสดงเป็นภาพแผ่นดิสก์อยู่ในซอง ข้อมูลภาพมี 23 รายการ กำหนดให้เป็นข้อมูลชนิดสตริงขนาด 7 ตัวอักษร อักษรตัวแรกจะเป็น M หรือ L โดย M แทน MoveTo และ L แทน LineTo อักษรตัวที่ 2 ถึง 4 เป็นค่าตำแหน่ง X อักษรตัวที่ 5 ถึง 7 เป็นค่าตำแหน่ง Y ซึ่งมีความหมายดังนี้

M100000 เลื่อน CP ไปที่ 100,000

L500000 ลากเส้นตรงจาก 100,000 ไปที่ 500,000

L500230 ลากเส้นตรงจาก 500,000 ไปที่ 500,230

L100230 ลากเส้นตรงจาก 500,230 ไปที่ 100,230

L100000 ลากเส้นตรงจาก 100,230 ไปที่ 100,000

โปรแกรมจะนำรายการข้อมูลภาพชุดนี้ไปสร้างเป็นรูปสี่เหลี่ยมผืนผ้าและแสดงที่จอภาพ ถ้าต้องการเพิ่มเติม หรือเปลี่ยนแปลงภาพก็สามารถทำได้โดยการเพิ่ม หรือเปลี่ยนแปลงรายการข้อมูลตามหลักการที่กล่าวแล้ว สำหรับโพรซีเยอร์ DisplayDrawing มีหน้าที่นำข้อมูลภาพมาแสดงที่จอภาพ โพรซีเยอร์นี้ไม่ต้องเปลี่ยนแปลงใดๆ เพราะสามารถจะนำข้อมูลภาพมาแสดงได้มากเท่าที่หน่วยความจำของคอมพิวเตอร์จะมีพอ

โปรแกรม 3.3

```
PROGRAM PictureFromVectorProgram;
USES Graph;
CONST MaxVector = 22;
   Disk : ARRAY[1..MaxVector] OF STRING[7] = (
        'M100000','L500000','L500230','L100230','L100000',
        'M115230','L115330','L485330','L485245','L475245','L475230',
        'M120327','L280327','L280250','L120250','L120327',
        'M100230','L100300','L115300',
        'M500230','L500300','L485300');
```

```
TYPE
   Pointer = ^Pixel;
   Pixel   = RECORD
               Visible : Boolean;
               X       : Real;
               Y       : Real;
               Next    : Pointer;
             END;
VAR
   VectorFile : Text;
   Points     : Pointer;
   GraphDriver,GraphMode : Integer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE ReadArray(VAR Points : Pointer);
VAR Last,Ptr     : Pointer;
    Command      : Char;
    Code,Count   : Integer;
    OneLine      : STRING[7];
    StringX,StringY : STRING[3];
BEGIN
   Points := Nil;
   FOR Count := 1 TO MaxVector DO
```

```
   BEGIN
      OneLine := Disk[Count];
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Command := OneLine[1];
         StringX := Copy(OneLine,2,3);
         StringY := Copy(OneLine,5,3);
         IF Command IN['m','M'] THEN Visible := False ELSE
         IF Command IN['l','L'] THEN Visible := True;
         Val(StringX,X,Code);
         Val(StringY,Y,Code);
      END;
      Last := Ptr;
   END;
   Last^.Next := Nil;
END;

PROCEDURE DisplayDrawing(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc  := Round(Graphics^.X);
      Yc  := Round(AspectRatio * Graphics^.Y);
      Yc  := GetMaxY - Yc;  {for turbo pascal}
```

```
      IF Graphics^.Visible THEN LineTo(Xc,Yc) ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;

BEGIN
   ReadArray(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DisplayDrawing(Points);
   ReadLn;
   CloseGraph
END.
```

3.2 การบันทึกภาพไว้ในไฟล์ข้อมูลภาพ

การบันทึกภาพไว้ในไฟล์ข้อมูลภาพ หมายถึง การนำเอาเฉพาะข้อมูลภาพมาบันทึกไว้ในไฟล์ ซึ่งเป็นการบันทึกแยกจากไฟล์โปรแกรม โดยในไฟล์โปรแกรมจะมีคำสั่งเกี่ยวกับการนำข้อมูลที่มีอยู่ในไฟล์ข้อมูลภาพมาจัดแสดงให้เป็นภาพที่ถูกต้อง ส่วนในไฟล์ข้อมูลภาพจะมีเฉพาะข้อมูลเกี่ยวกับภาพเท่านั้น จะเห็นได้ว่า การบันทึกภาพด้วยวิธีนี้อย่างน้อยจะต้องมีไฟล์ 2 ไฟล์ทำงานร่วมกัน ซึ่งจะมีผลให้แสดงภาพได้ช้ากว่าวิธีแรก ส่วนดีของการบันทึกภาพวิธีนี้ก็คือ โปรแกรมสามารถแสดงภาพได้ไม่จำกัดจำนวน โดยไม่ต้องเปลี่ยนแปลงส่วนใดในโปรแกรมเลย หมายความว่า ถ้าเราสร้างไฟล์ข้อมูลภาพขึ้นได้ โปรแกรมจะแสดงภาพนั้นได้ โปรแกรมสำเร็จรูปทางด้านกราฟิกส์และโปรแกรมเกมคอมพิวเตอร์หลายโปรแกรมเลือกใช้วิธีนี้

การบันทึกภาพไว้ในไฟล์ข้อมูลภาพ สามารถทำได้ทั้งการสร้างภาพกราฟิกส์แบบบิตแมพและภาพกราฟิกส์แบบเวกเตอร์ ซึ่งมีรายละเอียดดังต่อไปนี้

3.2.1 การบันทึกภาพแบบบิตแมพไว้ในไฟล์ข้อมูลภาพ สามารถทำได้หลายวิธี ในที่นี้จะเสนอวิธีการที่ทำได้สะดวกคือ การสร้างภาพด้วยตัวอักษรและบันทึกตัวอักษรเหล่านี้ไว้เป็นไฟล์ข้อมูลภาพ ซึ่งมีขั้นตอนวิธีดำเนินการดังนี้

1. ใช้เทอร์โบปาสคาลเอดิเตอร์ (ที่เราใช้สร้างโปรแกรมปาสคาล หรือเอ-ดิเตอร์อื่นๆ ก็ได้) สร้างภาพในลักษณะจุด ซึ่งอาจจะใช้เครื่องหมาย * หรือเครื่อง-หมายอื่นก็ได้ ตัวอย่างภาพแสดงในรูปที่ 3.1

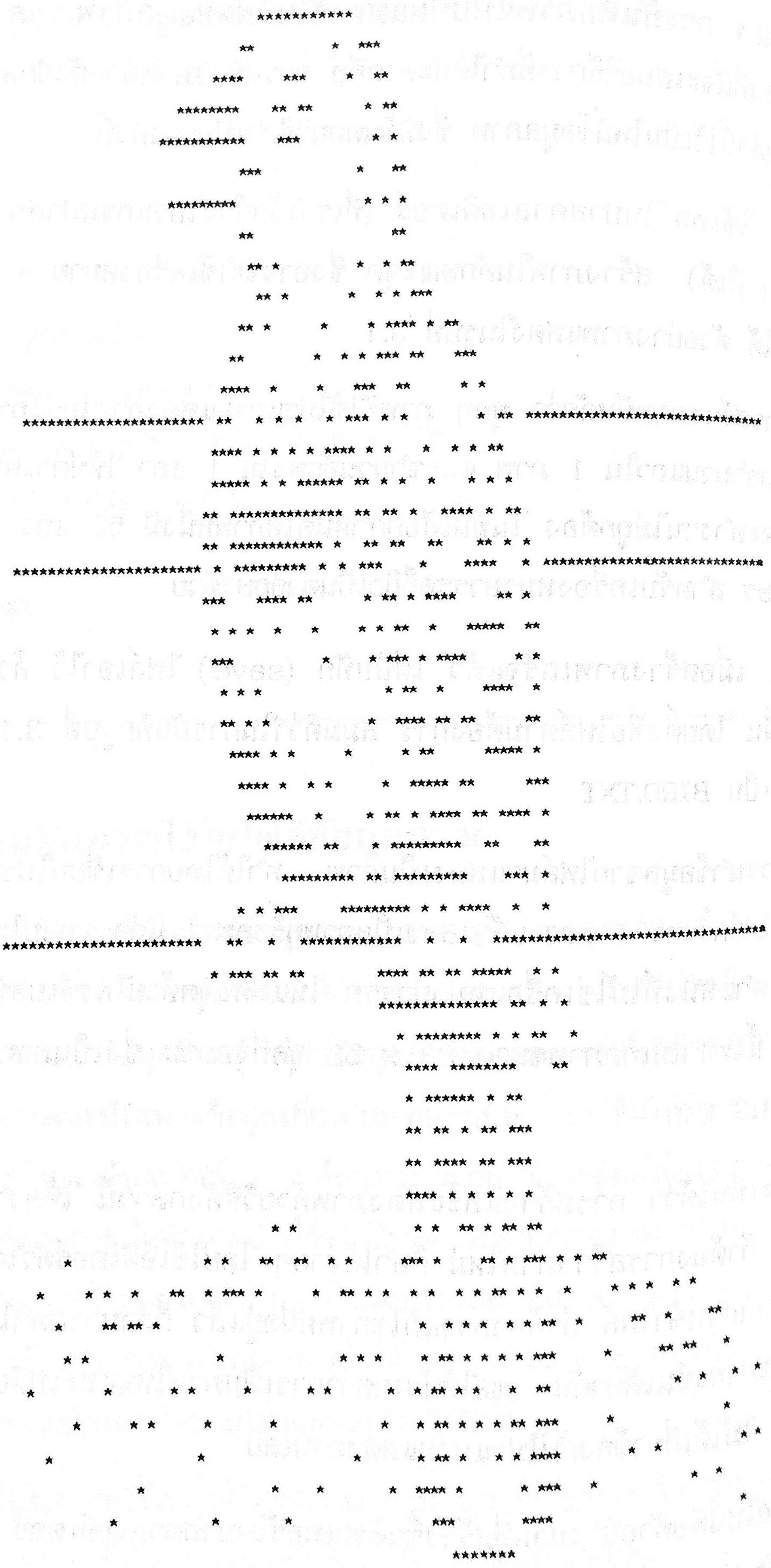
การสร้างภาพมีหลักว่า ทุกๆ ภาพที่ใช้โปรแกรมแสดงภาพโปรแกรมเดียวกันจะต้องกำหนดจำนวนแถวใน 1 ภาพ และจำนวนอักษรใน 1 แถว ให้มีค่าเท่ากัน มิฉะนั้นโปรแกรมจะทำงานไม่ถูกต้อง ในที่นี้เลือกกำหนดให้ภาพหนึ่งมี 52 แถว แต่ละแถวมี 88 ตัวอักษร สำหรับเครื่องหมายวรรคก็ถือเป็นตัวอักษรด้วย

2. เมื่อสร้างภาพเสร็จแล้ว ให้บันทึก (save) ไฟล์เอาไว้ ด้วยวิธีการของเอดิเตอร์นั้น โดยตั้งชื่อไฟล์ตามต้องการ สมมติว่าในการบันทึกรูปที่ 3.1 เราอาจจะตั้งชื่อไฟล์เป็น BIRD.TXT

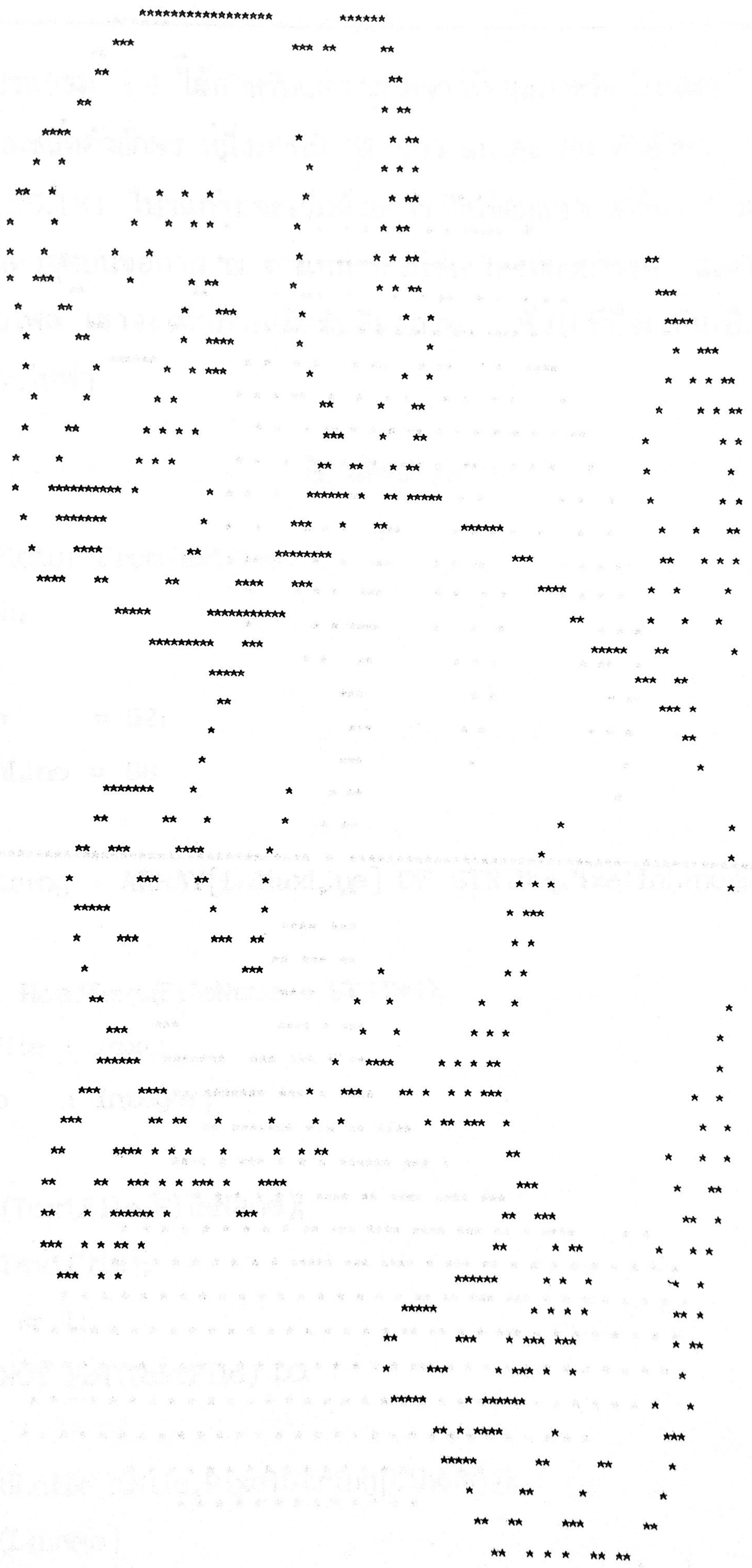
การนำข้อมูลจากไฟล์มาแสดงเป็นภาพ ทำได้โดยการเขียนโปรแกรมให้อ่านข้อมูลจากไฟล์ครั้งละ 1 แถว แล้วแสดงเป็นภาพครั้งละ 1 แถว จนจบไฟล์โดยมีเงื่อนไขว่า ณ ตำแหน่งที่ไม่ใช่เครื่องหมายวรรค ให้แสดงจุดด้วยโพรซีเยอร์ Putpixel ด้วยวิธีการนี้จะทำให้เกิดภาพขนาด 88 x 52 จุดที่จอภาพ ซึ่งเป็นภาพที่มีขนาดเล็กกว่ารูปที่ 1.3 มาก

จะเห็นได้ว่า การสร้างและแสดงภาพด้วยวิธีดังกล่าวนี้ ให้ความสะดวกมาก กล่าวคือ ถ้าต้องการสร้างภาพใหม่ ก็ทำได้ง่ายๆ โดยใช้เอดิเตอร์ตัวใดตัวหนึ่งสร้างภาพ แล้วบันทึกเข้าไฟล์ ถ้าต้องการแก้ไขภาพที่มีอยู่แล้ว ก็สามารถทำได้โดยใช้เอดิ-เตอร์แก้ไขภาพเช่นเดียวกัน ขอให้สังเกตว่าการเพิ่มภาพใหม่และเปลี่ยนแปลงแก้ไขภาพเดิม ไม่ได้เกี่ยวข้องกับโปรแกรมแสดงภาพเลย

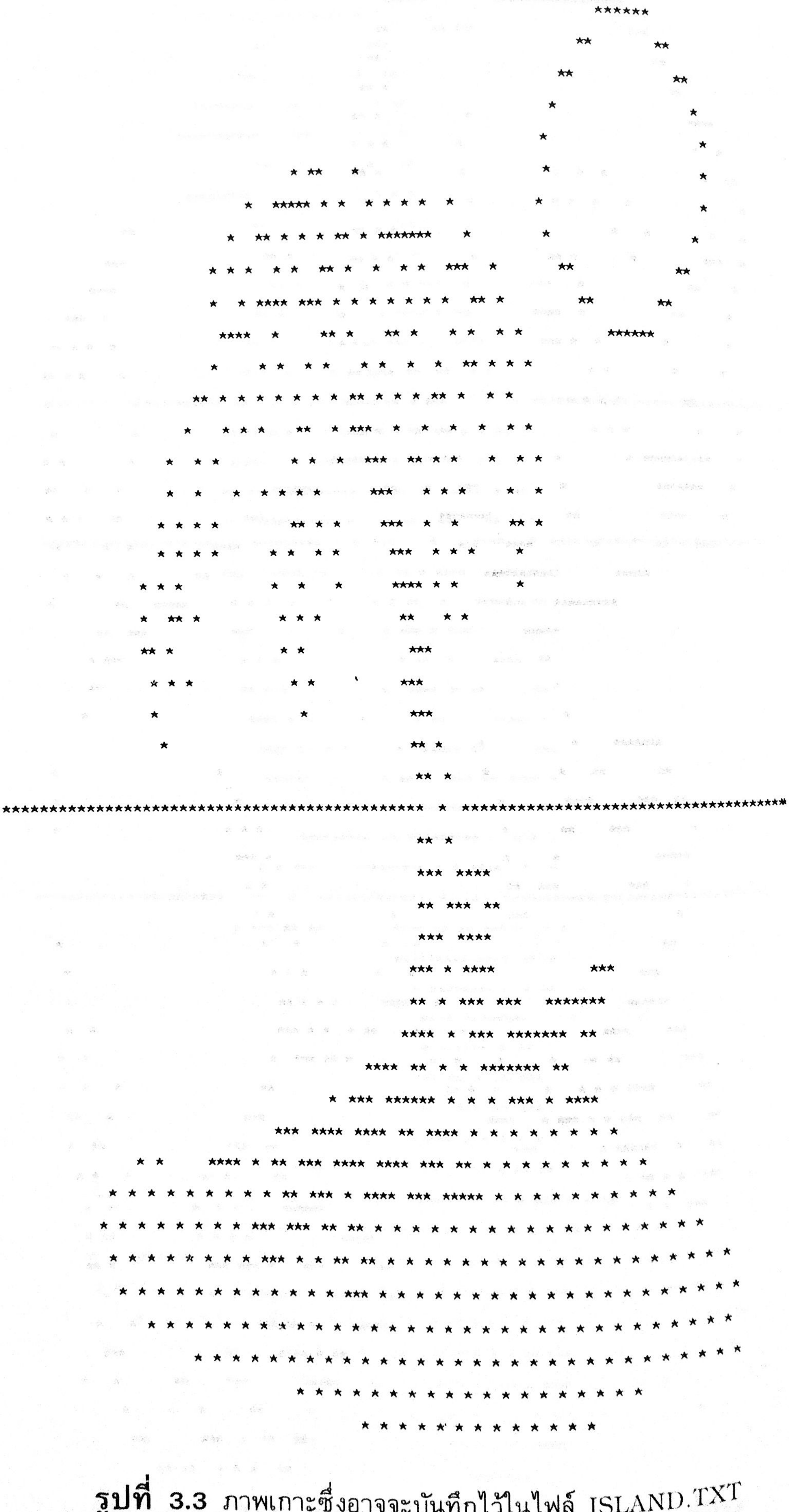
ต่อไปนี้คือตัวอย่างภาพที่สร้างขึ้นด้วยเทอร์โบปาสคาลเอดิเตอร์ ภาพแรกเป็นภาพลูกสุนัข ดังแสดงในรูปที่ 3.2 ภาพที่สองเป็นภาพเกาะ ดังแสดงในรูปที่ 3.3



รูปที่ 3.1 ภาพที่สร้างแบบบิตแมพโดยใช้เทอร์โบปาสคาลเอดิเตอร์



รูปที่ 3.2 ภาพลูกสุนัขซึ่งอาจจะบันทึกไว้ในไฟล์ POPPY.TXT



รูปที่ 3.3 ภาพเกาะซึ่งอาจจะบันทึกไว้ในไฟล์ ISLAND.TXT

ตัวอย่าง โปรแกรม 3.4 ใช้สำหรับแสดงภาพจากข้อมูลภาพที่อยู่ในไฟล์ ซึ่งเป็นข้อมูลภาพชนิดตัวอักษร หนึ่งภาพมี 52 แถว แถวละ 88 ตัวอักษร ในที่นี้คือไฟล์ BIRD.TXT โปรแกรมจะเริ่มด้วยการอ่านข้อมูลจากไฟล์มา 1 แถว แล้วแสดงจุดบนจอภาพ ณ ตำแหน่งที่ไม่ใช่เครื่องหมายวรรค และทำเช่นนี้ไปจนจบไฟล์ เราจะเห็นภาพเกิดขึ้นที่จอภาพ ซึ่งในที่นี้จะเห็นเป็นภาพนกเกาะสายไฟฟ้า

โปรแกรม 3.4

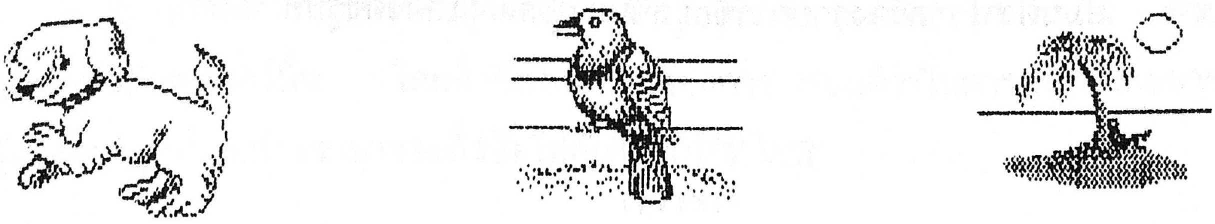
```
PROGRAM PictureFromTextFile;
USES Graph;
CONST
   MaxLine      = 52;
   PixelInLine  = 88;
VAR
   PixelString : ARRAY[1..MaxLine] OF STRING[PixelInLine];

PROCEDURE ReadText(FileName : STRING);
VAR TextFile : Text;
    LineNo    : Integer;
BEGIN
   Assign(TextFile,FileName);
   Reset(TextFile);
   LineNo := 1;
   WHILE NOT Eof(TextFile) DO
   BEGIN
      ReadLn(TextFile,PixelString[LineNo]);
      Inc(LineNo)
   END;
```

```
   Close(TextFile);
END;

PROCEDURE DisplayPixels(X,Y : Integer);
VAR GraphDriver,GraphMode                : Integer;
    ByteNo,ByteInLine,StartX,Line       : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   StartX := X;
   FOR Line := 1 TO MaxLine DO
   BEGIN
      ByteInLine := 1;
      FOR ByteNo := 1 TO PixelInLine DO
      BEGIN
         IF PixelString[Line,ByteNo] <> ' ' THEN PutPixel(X,Y,White);
         Inc(X);
      END;
      Inc(Y);  X := StartX;
   END;
END;

BEGIN
   ReadText('BIRD.TXT');
   DisplayPixels(100,100);
   ReadLn;
   CloseGraph
END.
```



รูปที่ 3.4 ภาพจากข้อมูลแบบบิตแมพที่เห็นบนจอภาพ

3.2.2 การบันทึกภาพแบบเวกเตอร์ไว้ในไฟล์ข้อมูลภาพ สามารถทำได้หลายวิธี ในที่นี้จะเสนอวิธีการที่ทำได้สะดวกคือ การสร้างภาพด้วยข้อความและบันทึกข้อความเหล่านี้ไว้เป็นไฟล์ข้อมูลภาพ ซึ่งมีขั้นตอนวิธีดำเนินการดังนี้

1. ใช้เทอร์โบปาสคาลเอดิเตอร์ (ที่เราใช้สร้างโปรแกรมปาสคาล หรือเอ–ดิเตอร์อื่นๆ ก็ได้) สร้างข้อความที่เป็นกฎของการสร้างภาพ ตามปกติเราสามารถตั้งกฎการสร้างภาพได้ตามใจชอบ เพียงแต่จะต้องให้สอดคล้องกับโปรแกรมแสดงภาพเป็นอย่างดี ในที่นี้จะกำหนดกฎของการสร้างภาพเพื่อเป็นตัวอย่างดังต่อไปนี้

(1) ข้อมูลแต่ละรายการเป็นข้อความที่มีความยาว 7 ตัวอักษร

(2) อักษรตัวแรกจะเป็น M, m, L หรือ l

โดย M และ m แทน MoveTo

L และ l แทน LineTo

(3) อักษรตัวที่ 2 ถึง 4 เป็นค่าตำแหน่ง X

อักษรตัวที่ 5 ถึง 7 เป็นค่าตำแหน่ง Y

(4) รายการข้อมูลแต่ละรายการ จะต้องจัดให้อยู่รายการละ 1 แถว โดยแต่ละแถวจะต้องเริ่มจากคอลัมน์ที่ 1 และจบที่คอลัมน์ 7 จากคอลัมน์ที่ 8 เป็นต้นไป อาจจะใช้สำหรับอธิบายความหมายของรายการข้อมูลนั้นได้ดังตัวอย่างต่อไปนี้

M100000 เลื่อนซีพีไปที่ 100,000

L500000 ลากเส้นตรงจาก 100,000 ไปที่ 500,000

L500230 ลากเส้นตรงจาก 500,000 ไปที่ 500,230

สมมติว่าเราสร้างรายการข้อมูลขึ้นมาชุดหนึ่งแสดงดังรูปที่ 3.5

```
M100000 envelope
L500000
L500230
L100230
L100000
M115030 diskette
L115330
L485330
L485245 protect
L475245
L475230
M120327 label
L280327
L280250
L120250
L120327
M100230 left up
L100300
L115300
M500230 right up
L500300
L485300
```

รูปที่ 3.5 รายการข้อมูลสร้างภาพแบบเวกเตอร์ซึ่งบันทึกไว้ในไฟล์ DISK.VCT

2. เมื่อสร้างรายการข้อมูลเสร็จแล้ว ให้บันทึก (save) ไฟล์เอาไว้ ด้วยวิธีการของเอดิเตอร์นั้น โดยตั้งชื่อไฟล์ตามต้องการ สมมติว่าในการบันทึกรายการข้อมูลตามรูปที่ 3.5 เราอาจจะตั้งชื่อไฟล์เป็น DISK.VCT

การนำข้อมูลจากไฟล์มาแสดงเป็นภาพ ทำได้โดยการเขียนโปรแกรมให้อ่านข้อมูลจากไฟล์ครั้งละ 1 แถว แล้วเปลี่ยนรายการข้อมูลนั้นแสดงเป็นภาพ โดยมีเงื่อนไขดังกล่าวแล้วในข้อ (4) เมื่อจบไฟล์จะเกิดภาพหนึ่งภาพขึ้นที่จอภาพ

จะเห็นได้ว่า การสร้างและแสดงภาพด้วยวิธีดังกล่าวนี้ ให้ความสะดวกมาก กล่าวคือ ถ้าต้องการสร้างภาพใหม่ ก็ทำได้ง่ายๆ โดยการใช้เอดิเตอร์ตัวใดตัวหนึ่งสร้างรายการข้อมูลภาพ แล้วบันทึกเข้าไฟล์ ถ้าต้องการแก้ไขภาพที่มีอยู่แล้ว ก็สามารถทำได้ โดยใช้เอดิเตอร์แก้ไขรายการข้อมูลเช่นเดียวกัน ขอให้สังเกตว่าการเพิ่มภาพใหม่ และเปลี่ยนแปลงแก้ไขภาพเดิม ไม่ได้เกี่ยวข้องกับโปรแกรมแสดงภาพเลย

ตัวอย่าง โปรแกรม 3.5 ใช้สำหรับแสดงภาพจากข้อมูลภาพที่อยู่ในไฟล์ ซึ่งเป็นข้อมูลภาพชนิดข้อความ ในที่นี้คือข้อมูลจากไฟล์ DISK.VCT โปรแกรมจะเริ่มด้วยการอ่านข้อมูลจากไฟล์มา 1 แถว แล้วดำเนินการตามเงื่อนไขที่กำหนดไว้ คือ เลื่อนซีพีหรือลากเส้นตรง และทำเช่นนี้ไปจนจบไฟล์ เราจะเห็นภาพเกิดขึ้นที่จอภาพ ซึ่งในที่นี้จะเห็นเป็นภาพดิสก์อยู่ในซอง

โปรแกรม 3.5

```
PROGRAM PictureFromVectorFile;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel    = RECORD
                Visible : Boolean;
                X          : Real;
```

```
              Y         : Real;
              Next      : Pointer;
           END;
VAR
   VectorFile : Text;
   Points     : Pointer;
   GraphDriver,GraphMode : Integer;

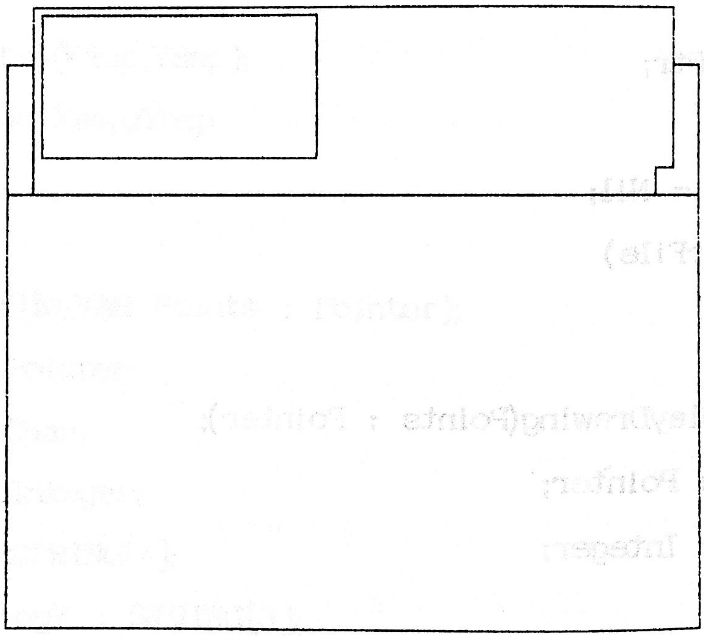
FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE ReadFile(VAR Points : Pointer);
VAR Last,Ptr : Pointer;
    Command  : Char;
    Code     : Integer;
    OneLine  : STRING[7];
    StringX,StringY : STRING[3];
BEGIN
   Assign(VectorFile,'DISK.VCT');
   Reset(VectorFile);
   Points := Nil;
   WHILE NOT Eof(VectorFile) DO
   BEGIN
      ReadLn(VectorFile,OneLine);
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
```

```
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Command := OneLine[1];
         StringX := Copy(OneLine,2,3);
         StringY := Copy(OneLine,5,3);
         IF Command IN['m','M'] THEN Visible := False ELSE
         IF Command IN['l','L'] THEN Visible := True;
         Val(StringX,X,Code);
         Val(StringY,Y,Code);
      END;
      Last := Ptr;
   END;
   Last^.Next := Nil;
   Close(VectorFile)
END;

PROCEDURE DisplayDrawing(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc  := Round(Graphics^.X);
      Yc  := Round(AspectRatio * Graphics^.Y);
      Yc  := GetMaxY - Yc;  {for turbo pascal}
      IF Graphics^.Visible THEN LineTo(Xc,Yc) ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;
```

```
BEGIN
   ReadFile(PointS);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DisplayDrawing(Points);
   ReadLn;
   CloseGraph
END.
```



รูปที่ 3.6 ภาพจากข้อมูลแบบเวกเตอร์ที่ได้จากโปรแกรม 3.5

3.3 การบันทึกภาพไว้ในโปรแกรมและในไฟล์ข้อมูลภาพ

การบันทึกภาพไว้ในโปรแกรมและในไฟล์ข้อมูลภาพ หมายถึง การบันทึกข้อมูลภาพบางส่วนไว้ในโปรแกรม และบันทึกข้อมูลภาพอีกส่วนหนึ่งไว้ในไฟล์ข้อมูลภาพ ข้อมูลภาพที่อยู่ในโปรแกรมจะเป็นข้อมูลที่ไม่มีการเปลี่ยนแปลง ส่วนข้อมูลภาพในไฟล์เป็นข้อมูลที่เปลี่ยนแปลงได้หรือเป็นข้อมูลที่โปรแกรมจะเลือกนำมาใช้ในโอกาสต่างๆกัน โปรแกรมสำเร็จรูปทางด้านกราฟิกส์และเกมคอมพิวเตอร์ส่วนใหญ่เลือกใช้วิธีนี้ เช่น จัดให้

ข้อมูลภาพเกี่ยวกับฉากของภาพอยู่ในโปรแกรม ส่วนข้อมูลภาพที่เป็นตัวละครจัดบันทึกไว้ในไฟล์ ทำให้โปรแกรมสามารถแสดงภาพตัวละครที่มีลักษณะแตกต่างกันได้หลายลักษณะที่ฉากเดิม

การบันทึกภาพด้วยวิธีนี้สามารถใช้ได้กับภาพทุกแบบคือ แบบบิตแมพ แบบเวกเตอร์ และแบบผสมบิตแมพกับเวกเตอร์

ตัวอย่าง โปรแกรม 3.6 แสดงการบันทึกข้อมูลภาพบางส่วนไว้ในโปรแกรมและบันทึกข้อมูลภาพอีกส่วนหนึ่งไว้ในไฟล์ข้อมูลภาพในที่นี้กำหนดให้ข้อมูลภาพแบบเวกเตอร์ ซึ่งเป็นภาพแผ่นดิสก์อยู่ในซอง เป็นข้อมูลอยู่ในโปรแกรม ส่วนข้อมูลแบบบิตแมพซึ่งเป็นรูปนกจัดบันทึกอยู่ในไฟล์ BIRD.TXT โปรแกรมจะเริ่มด้วยการแสดงดิสก์ที่ไม่มีตรา เมื่อกดคีย์ Enter โปรแกรมจะอ่านข้อมูลจากไฟล์ BIRD.TXT มาแสดงที่ซองดิสก์และแผ่นดิสก์ ถ้าต้องการเปลี่ยนเป็นตราอื่นก็สามารถทำได้โดยการสร้างภาพใหม่ และบันทึกไว้ในไฟล์ใหม่ด้วยวิธีการตามข้อ 3.2.1 แล้วเปลี่ยนชื่อไฟล์ข้อมูลภาพในโปรแกรมจาก BIRD.TXT ให้เป็นไฟล์ใหม่เช่น PUPPY.TXT เราจะได้ดิสก์ตราใหม่ตามต้องการ

โปรแกรม 3.6

```
PROGRAM PictureFromVectorProgramBitmapFile;
USES Graph;
CONST MaxVector = 23;
  Disk : ARRAY[1..MaxVector] OF STRING[7] = (
       'M100000','L500000','L500230','L500230','L100230','L100000',
       'M115230','L115330','L485330','L485245','L475245','L475230',
       'M120327','L280327','L280250','L120250','L120327',
       'M100230','L100300','L115300',
       'M500230','L500300','L485300');
```

```
TYPE
   Pointer = ^Pixel;
   Pixel   = RECORD
                  Visible : Boolean;
                  X       : Real;
                  Y       : Real;
                  Next    : Pointer;
               END;
VAR
   VectorFile : Text;
   Points     : Pointer;
   GraphDriver,GraphMode : Integer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE ReadArray(VAR Points : Pointer);
VAR Last,Ptr     : Pointer;
    Command      : Char;
    Code,Count : Integer;
    OneLine      : STRING[7];
    StringX,StringY : STRING[3];
BEGIN
   Points := Nil;
   FOR Count := 1 TO MaxVector DO
```

```
    BEGIN
      OneLine := Disk[Count];
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
        Command := OneLine[1];
        StringX := Copy(OneLine,2,3);
        StringY := Copy(OneLine,5,3);
        IF Command IN['m','M'] THEN Visible := False ELSE
        IF Command IN['l','L'] THEN Visible := True;
        Val(StringX,X,Code);
        Val(StringY,Y,Code);
      END;
      Last := Ptr;
    END;
    Last^.Next := Nil;
END;

PROCEDURE DisplayDrawing(Points : Pointer);
VAR Graphics  : Pointer;
    Xc,Yc     : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
     Xc   := Round(Graphics^.X);
     Yc   := Round(AspectRatio * Graphics^.Y);
```

```
    Yc  := GetMaxY - Yc;  {for turbo pascal}
    IF Graphics^.Visible THEN LineTo(Xc,Yc) ELSE MoveTo(Xc,Yc);
    Graphics := Graphics^.Next;
  END;
END;

{ ----------  read file and display bitmap picture  ------- }

PROCEDURE PictureFromTextFile;
CONST
  MaxLine     = 52;
  PixelInLine = 88;
VAR
  PixelString : ARRAY[1..MaxLine] OF STRING[PixelInLine];

PROCEDURE ReadText(FileName : STRING);
VAR TextFile  : Text;
    LineNo    : Integer;
BEGIN
  Assign(TextFile,FileName);
  Reset(TextFile);
  LineNo := 1;
  WHILE NOT Eof(TextFile) DO
  BEGIN
    ReadLn(TextFile,PixelString[LineNo]);
    Inc(LineNo)
  END;
  Close(TextFile);
END;
```

```
PROCEDURE DisplayPixels(X,Y : Integer);
VAR GraphDriver,GraphMode                : Integer;
    ByteNo,ByteInLine,StartX,Line       : Integer;
BEGIN
   StartX := X;
   FOR Line := 1 TO MaxLine DO
   BEGIN
      ByteInLine := 1;
      FOR ByteNo := 1 TO PixelInLine DO
      BEGIN
         IF PixelString[Line,ByteNo] <> ' ' THEN PutPixel(X,Y,Blue);
         Inc(X);
      END;
      Inc(Y);  X := StartX;
   END;
END;

BEGIN
   ReadText('BIRD.TXT');
   IF GraphDriver = 7 THEN {Hercules}
   BEGIN
      DisplayPixels(158,105);
      DisplayPixels(378,280);
   END
   ELSE
   BEGIN {VGA}
      DisplayPixels(158,160);
      DisplayPixels(378,380);
   END
END;
```

```
{ ---------------- end of PictureFromTextFile ----------------}

BEGIN { MAIN }
   ReadArray(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DisplayDrawing(Points);
   ReadLn;
   PictureFromTextFile;
   ReadLn;
   CloseGraph
END.
```

3.4 การบันทึกข้อมูลภาพและแสดงภาพตามแบบของเทอร์โบปาส-คาล

การบันทึกข้อมูลภาพเอาไว้ในไฟล์ สามารถทำได้หลายรูปแบบและหลายวิธีการ ซึ่งเราได้เห็นตัวอย่างมาแล้วจากหัวข้อที่ผ่านมา ตามปกติโปรแกรมกราฟิกส์ต่างๆ จะมีวิธีดำเนินการเกี่ยวกับรูปแบบของข้อมูลภาพแตกต่างกัน สำหรับโปรแกรมที่มีข้อมูลภาพเป็นแบบเดียวกันจะสามารถใช้ไฟล์ภาพร่วมกันได้ มีบางโปรแกรมสามารถเปลี่ยนรูปแบบข้อมูลภาพจากโปรแกรมอื่นที่แตกต่างกันให้เป็นรูปแบบของตนเองได้ ทำให้สามารถนำไฟล์ภาพของโปรแกรมอื่นมาใช้ในโปรแกรมของตนได้ โดยไม่ต้องเสียเวลาสร้างขึ้นใหม่ เทอร์โบปาสคาลก็เช่นเดียวกับโปรแกรมกราฟิกส์อื่นๆ คือ มีวิธีดำเนินการกับกราฟิกส์ในแบบของตนเองซึ่งจะได้กล่าวถึงรายละเอียดดังต่อไปนี้

3.4.1 การบันทึกภาพจากจอภาพ ถ้าให้เทอร์โบปาสคาลบันทึกภาพจากจอภาพ รูปแบบของข้อมูลภาพจะเป็นแบบบิตแมพและรายการข้อมูลภาพจะเป็นรหัส ASCII ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 3.7 ทำงาน 2 อย่างคือ แสดงภาพจากไฟล์ข้อมูลภาพที่เป็นไฟล์ตัวอักษร ซึ่งมีขนาดภาพเท่ากับ 88 x 52 ตัวอักษร ในโปรแกรมนี้ใช้ข้อมูลจากไฟล์ BIRD.TXT และเมื่อกดคีย์ Enter โปรแกรมจะบันทึกข้อมูลภาพจากจอภาพไว้ในไฟล์ BIRD.GTP

โปรแกรม 3.7

```
PROGRAM SavePictureOnScreen;
USES Graph;
CONST
  MaxLine      = 52;
  PixelInLine = 88;
VAR
  PixelString : ARRAY[1..MaxLine] OF STRING[PixelInLine];

PROCEDURE ReadText(FileName : STRING);
VAR TextFile : Text;
    LineNo   : Integer;
BEGIN
  Assign(TextFile,FileName);
  Reset(TextFile);
  LineNo := 1;
  WHILE NOT Eof(TextFile) DO
  BEGIN
    ReadLn(TextFile,PixelString[LineNo]);
    Inc(LineNo)
  END;
  Close(TextFile);
END;
```

```
PROCEDURE DisplayPixels(X,Y : Integer);
VAR GraphDriver,GraphMode                : Integer;
    ByteNo,ByteInLine,StartX,Line        : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   StartX := X;
   FOR Line := 1 TO MaxLine DO
   BEGIN
      ByteInLine := 1;
      FOR ByteNo := 1 TO PixelInLine DO
      BEGIN
         IF PixelString[Line,ByteNo] <> ' ' THEN PutPixel(X,Y,White);
         Inc(X);
      END;
      Inc(Y);   X := StartX;
   END;
END;


PROCEDURE SavePicture(FileName : STRING);
{save picture on screen in Turbo Pascal Graphic format (.GTP)}
VAR  PictureFile    : File;
     PicturePointer : Pointer;
     MemUsed        : Word;    {byte of memory used in the picture}
     X1,Y1,X2,Y2    : Integer;
BEGIN
   X1 := 8; Y1 := 0; X2 := 95; Y2 := 51;
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
```

```
  GetImage(X1,Y1,X2,Y2,PicturePointer^);
  WHILE MemUsed MOD 128 <> 0 DO MemUsed := MemUsed + 1;
  MemUsed := MemUsed DIV 128; {number of record}
  Assign(PictureFile,FileName);
  ReWrite(PictureFile);
  BlockWrite(PictureFile,PicturePointer^,MemUsed);
  Close(PictureFile);
END;

BEGIN
  ReadText('BIRD.TXT');
  DisplayPixels(8,0);
  ReadLn;
  SavePicture('BIRD.GTP');
  CloseGraph
END.
```

โปรแกรม 3.7 จะแสดงภาพด้วยโพรซีเยอร์ DisplayPixels ตามวิธีการที่ได้กล่าวมาแล้วในข้อ 3.2.1 สำหรับการบันทึกภาพใช้โพรซีเยอร์ SavePicture ซึ่งมีขั้นตอนการทำงานดังต่อไปนี้

1. กำหนดขอบเขตของภาพที่จะบันทึกให้อยู่ในกรอบสี่เหลี่ยม ซึ่งมี X1,Y1 เป็นจุดมุมบนซ้าย และมี X2,Y2 เป็นจุดมุมล่างขวาของกรอบ ข้อมูลภาพภายในกรอบนี้เท่านั้นที่จะถูกบันทึกไว้ในไฟล์ ตามตัวอย่างนี้กำหนดให้ X1 = 8, Y1 = 0, X2 = 95 และ Y2 = 51 ซึ่งจะเป็นกรอบขนาด 88 x 52 พิกเซล ล้อมรอบภาพนกบนจอภาพได้พอดี

2. หาขนาดของหน่วยความจำซึ่งอยู่ภายในกรอบของภาพที่กำหนดไว้ตามข้อ 1 ด้วยฟังก์ชัน ImageSize ซึ่งมีรูปแบบดังนี้

ImageSize (X1, Y1, X2, Y2 : Integer)

โดย X1,Y1 คือมุมบนซ้ายของกรอบภาพ

X2,Y2 คือมุมล่างขวาของกรอบภาพ

สเตตเมนต์ในโปรแกรมคือ

MemUsed := ImageSize(X1,Y1,X2,Y2)

หมายถึงให้เก็บขนาดหน่วยความจำของภาพไว้ใน MemUsed

3. จองหน่วยความจำตามขนาดที่หาได้จากฟังก์ชัน ImageSize ในข้อ 2 ด้วยไพรซีเยอร์ **GetMem** ซึ่งมีรูปแบบดังนี้

GetMem (VAR P : Pointer; Size : Word)

โดย P คือตำแหน่งของหน่วยความจำที่จอง

Size คือขนาดของหน่วยความจำที่จอง

สเตตเมนต์ในโปรแกรมคือ

GetMem(PicturePointer,MemUsed)

หมายถึงหน่วยความจำที่จองมีขนาดเท่ากับ MemUsed เริ่มต้นที่ตำแหน่ง PicturePionter

4. อ่านภาพจากกรอบที่กำหนดไว้บนจอภาพ เข้ามาในหน่วยความจำที่จองไว้ด้วยไพรซีเยอร์ **GetImage** ซึ่งมีรูปแบบดังนี้

GetImage (X1,Y1, X2,Y2 : Integer; VAR BitMap)

โดย X1,Y1 คือมุมบนซ้ายของกรอบภาพ

X2,Y2 คือมุมล่างขวาของกรอบภาพ

BitMap คือหน่วยความจำสำหรับเก็บภาพ

สเตตเมนต์ในโปรแกรมคือ

GetImage(X1,Y1, X2,Y2,PicturePointer^)

หมายถึง ให้เก็บภาพที่อยู่ในกรอบ X1,Y1, X2,Y2 เอาไว้ใน PicturePointer^

5. แบ่งหน่วยความจำที่เก็บภาพออกเป็นกลุ่ม ให้มีขนาดกลุ่มละ 128 ไบต์ ด้วยสเตตเมนต์ต่อไปนี้

```
MemUsed := MemUsed DIV 128
```

จำนวนกลุ่มที่ได้จะใช้เป็นค่าจำนวนเรคอร์ดเพื่อการบันทึกข้อมูลในสเตตเมนต์ต่อไปนี้

```
BlockWrite(PictureFile,PicturePointer^,MemUsed)
```

ในกรณีที่ค่าหน่วยความจำหารด้วย 128 ไม่ลงตัว เราจำเป็นจะต้องเพิ่มข้อมูลภาพซึ่งเป็นที่ว่างเข้าไปอีกจนกระทั่ง 128 หารได้ลงตัว โดยใช้สเตตเมนต์ต่อไปนี้

```
WHILE MemUsed MOD 128 <> 0 DO MemUsed := MemUsed + 1
```

6. บันทึกข้อมูลภาพเข้าไฟล์ด้วยสเตตเมนต์ต่อไปนี้

```
Assign(PictureFile,FileName);
ReWrite(PictureFile);
BlockWrite(PictureFile,PicturePointer^,MemUsed);
Close(PictureFile)
```

โปรแกรมนี้กำหนดให้ไฟล์ทั้งสองมีชื่อเดียวกัน แต่ชนิดของไฟล์ต่างกันคือ ไฟล์ใหม่มีชนิดเป็น GTP ซึ่งหมายถึง ไฟล์กราฟิกส์แบบเทอร์โบปาสคาล (Turbo Pascal Graphics File)

3.4.2 การแสดงภาพจากไฟล์ข้อมูลภาพ เทอร์โบปาสคาลสามารถนำข้อมูลภาพจากไฟล์ที่ได้จัดเก็บไว้ด้วยวิธีการในข้อ 3.4.1 มาแสดงที่จอภาพได้อย่างรวดเร็ว ด้วยวิธีการง่ายๆ ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 3.8 แสดงภาพจากไฟล์ข้อมูลภาพที่ได้จัดเก็บไว้ตามแบบของเทอร์โบปาสคาล ในตัวอย่างนี้เป็นภาพจากไฟล์ BIRD.GTP

โปรแกรม 3.8

```
PROGRAM PictureFromGtpFile;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile : File;
BEGIN
   Assign(PictureFile,'BIRD.GTP');
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut);
   Readln;
   CloseGraph;
END.
```

หลักการทำงานของโปรแกรม 3.8 เป็นดังนี้

1. เปิดไฟล์ข้อมูลภาพด้วยสเตตเมนต์ต่อไปนี้

```
Assign(PictureFile,'BIRD.GTP');
Reset(PictureFile)
```

2. จองหน่วยความจำไว้เท่ากับจำนวนข้อมูลที่มีในไฟล์ด้วยสเตตเมนต์ต่อไปนี้

```
GetMem(PicturePointer,FileSize(PictureFile))
```

3. อ่านข้อมูลจากไฟล์เข้ามาในหน่วยความจำด้วยสเตตเมนต์ต่อไปนี้

```
BlockRead(PictureFile,PicturePointer^,
          FileSize(PictureFile))
```

4. แสดงภาพที่มีอยู่ในหน่วยความจำด้วยไพรซีเยอร์ **PutImage** ซึ่งมีรูปแบบดังนี้

```
PutImage (X, Y : Integer; VAR BitMap; BitBit : Word)
```

โดย X,Y คือมุมบนซ้ายของกรอบภาพ

BitMap คือข้อมูลภาพ

BitBit คือลักษณะของภาพที่แสดงซึ่งกำหนดเป็นคอนสแตนต์ไว้ดังต่อไปนี้

```
CONST
    NormalPut   =  0;
    XORPut      =  1;
    ORPut       =  2;
    ANDPut      =  3;
    NOTPut      =  4;
```

สเตตเมนต์ที่ใช้ในโปรแกรมคือ

```
PutImage(0,0,PicturePointer^,NormalPut)
```

หมายถึงให้แสดงภาพลักษณะปกติที่มุมบนซ้ายของจอภาพ

ข้อสังเกต (1) โปรแกรม 3.8 นี้ สามารถใช้กับไฟล์ข้อมูลภาพของเทอร์โบปาสคาลที่มีการบันทึกข้อมูลภาพในรูปแบบของไฟล์ GTP ได้ทุกไฟล์

(2) โปรแกรมอาจจะต้องใช้ยูนิต GrDriver ด้วยคือ

```
USES Graph,GrDriver;
```

วิธีสร้างยูนิต GrDriver ดูได้จากภาคผนวก ข.

3.4.3 ส่วนประกอบของข้อมูลภาพในไฟล์ เมื่อเราสั่งให้เทอร์โบปาสคาลนำภาพที่ปรากฏอยู่บนจอภาพมาบันทึกไว้ในไฟล์ เทอร์โบปาสคาลจะเก็บข้อมูลภาพเหล่านั้นไว้ในรูปของรหัส ASCII ซึ่งถ้าเราสั่งให้ DOS แสดงข้อมูลนั้น เช่น ใช้คำสั่งดังต่อไปนี้

TYPE BIRD.GTP

บนจอภาพจะปรากฏตัวอักษรและสัญลักษณ์แปลกๆ ซึ่งตัวอักษรเหล่านี้ได้มาจากการเปลี่ยนรหัส ASCII ของข้อมูลที่มีอยู่ในไฟล์ให้เป็นตัวอักษรหรือสัญลักษณ์ของรหัสนั้น

เราสามารถดูข้อมูลที่เป็นรหัส ASCII จากไฟล์ข้อมูลภาพของเทอร์โบปาสคาลได้ด้วยโปรแกรมต่อไปนี้

ตัวอย่าง โปรแกรม 3.9 ใช้แสดงข้อมูลภาพจากไฟล์ภาพของเทอร์โบปาสคาล โปรแกรมจะแสดงรหัส ASCII เป็นเลขฐานสิบ ซึ่งจะมีค่าตั้งแต่ 0 ถึง 255 โปรแกรมนี้ใช้ได้กับไฟล์ที่มีขนาดตั้งแต่ 577 ไบต์ขึ้นไป สำหรับไฟล์ที่มีขนาดเล็กกว่านี้ ให้แก้ไขค่า FileSize ให้เท่ากับ หรือน้อยกว่าค่าของขนาดไฟล์นั้น ขนาดไฟล์ดูได้ด้วยคำสั่ง DIR ของ DOS

โปรแกรม 3.9

```
PROGRAM DecFromGtpFile;
USES Graph;
CONST FileSize = 577;
TYPE Pixel        = ARRAY[1..FileSize] OF Byte;
     PixelRec   = RECORD
                        PictureItem : Pixel;
                      END;
VAR PixelArray   : Pixel;
```

```
   PixelRecord : PixelRec;

PROCEDURE ReadPictureFile;
VAR PictureFile : File OF PixelRec;
    DecNo       : Integer;
BEGIN
   Assign(PictureFile,'BIRD.GTP');
   Reset(PictureFile);
   Read(PictureFile,PixelRecord);
   Close(PictureFile);
   PixelArray := PixelRecord.PictureItem;
   FOR DecNo := 1 TO FileSize DO
   Write(PixelArray[DecNo],' ')
END;

BEGIN
   ReadPictureFile;
   ReadLn
END.
```

ผลที่ได้จากการรันโปรแกรม 3.9 ซึ่งเป็นข้อมูลจากไฟล์ BIRD.GTP ที่บันทึกจากจอภาพระบบเฮอร์คิวลิส แสดงดังรูปที่ 3.7

เทอร์โบปาสคาลแบ่งข้อมูลภาพออกเป็น 2 ส่วนคือ ส่วนที่เป็นค่าความกว้างและค่าความสูงของภาพ กับส่วนที่เป็นข้อมูลภาพ ซึ่งมีรายละเอียดดังนี้

ไบต์ที่ 1 และ 2 เป็นค่าความกว้างของภาพ

ไบต์ที่ 3 และ 4 เป็นค่าความสูงของภาพ

ไบต์ที่ 5 และ 6 สงวนไว้สำหรับเทอร์โบปาสคาล

ตั้งแต่ไบต์ที่ 7 ไปจนจบไฟล์เป็นข้อมูลภาพ

87 0 51 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 15 254 0 0 0 0 0 0 0 0 0 48 9 192 0 0 0 0
0 0 0 1 225 194 96 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 127 241 192 80 0 0 0 0 0
0 0 0 56 1 24 0 0 0 0 0 0 0 63 224 0 168 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2
6 1 44 0 0 0 0 0 0 0 0 13 2 87 0 0 0 0 0 0 0 0 52 16 189 96 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 242 17 204 20 0 0 0 0 63 255 255 101 75 168 22 255 255 255 252 0
0 1 234 175 81 43 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 127 254 218 122 192 0 0 0
0 0 3 127 250 108 198 160 0 0 0 127 255 254 191 234 226 60 191 255 255 254 0 0
0 0 0 0 0 0 0 0 0 0 0 1 82 33 89 31 48 0 0 0 0 0 1 192 16 186 240 80 0 0 0 0 0 0
168 0 44 135 144 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 122 130 11 7 208 0 0 0 0 0 0
61 40 39 216 56 0 0 0 0 0 0 31 144 181 237 232 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 255 111 248 232 0 0 0 0 0 0 43 135 253 94 72 0 0 0 255 255 254 100 255 242 7
9 255 255 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 63 255 106 0 0 0 0 0 0 0 0 11
253 89 0 0 0 0 0 0 0 0 7 191 198 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 214 224
0 0 0 0 0 0 0 0 6 246 224 0 0 0 0 0 0 0 0 7 170 160 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 5 25 148 29 13 117 96 0 0 8 82 50 232 38 165 43 82 21 40 0 2 27 64 0 0 4 16
9 89 6 67 0 0 0 0 0 0 0 0 0 0 0 0 16 16 40 32 80 37 168 152 0 36 72 0 130 130 4
0 69 170 220 16 0 128 4 0 4 0 2 5 162 188 4 128 40 0 0 0 0 0 0 0 0 0 0 0 0 8 1 1
0 1 192 120 0 0 32 0 0 0 0 0 0 63 128 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

รูปที่ 3.7 ข้อมูลภาพจากไฟล์ BIRD.GTP ที่ได้จากการรันโปรแกรม 3.9

1. ค่าความกว้างและค่าความสูงของภาพ มีความหมายดังต่อไปนี้

เนื่องจาก 1 ไบต์ของรหัส ASCII เก็บค่าสูงสุดได้ไม่เกิน 255 ซึ่งไม่เพียงพอที่จะเก็บค่าความกว้างหรือค่าความสูงของภาพที่มีขนาดใหญ่ได้ จึงจำเป็นต้องใช้ 2 ไบต์ โดยมีเงื่อนไขดังนี้คือ

ไบต์ที่ 1 เป็นค่าความกว้างของภาพที่มีขนาดตั้งแต่ 0 ถึง 255

ไบต์ที่ 2 เป็นรหัสซึ่งกำหนดไว้ดังนี้ คือ เลข 0 แทนความกว้าง 0

1 แทนความกว้าง 256

2 แทนความกว้าง 512

3 แทนความกว้าง 768

เรียงลำดับเช่นนี้ต่อไปเรื่อยๆ

ค่าความกว้างของภาพจะหาได้จากผลบวกของค่าในไบต์ที่ 1 ค่าในไบต์ที่ 2 และค่า 1 (เนื่องจากเทอร์โบปาสคาลกำหนดจุดเริ่มต้นที่ 0,0)

ตัวอย่าง

ไบต์ที่ 1	ไบต์ที่ 2	ความกว้างของภาพ(พิกเซล)
87	0	87 + 0 + 1 = 88
87	1	87 + 256 + 1 = 344
255	0	255 + 0 + 1 = 256
91	1	91 + 256 + 1 = 348
207	2	207 + 512 + 1 = 720

สำหรับค่าความสูงของภาพก็มีหลักการเช่นเดียวกันคือ

ไบต์ที่ 3 เป็นค่าความสูงของภาพที่มีขนาดตั้งแต่ 0 ถึง 255

ไบต์ที่ 4 เป็นรหัสซึ่งกำหนดไว้เช่นเดียวกับค่าความกว้างดังที่ได้กล่าวมาแล้ว

2. ข้อมูลภาพมีความหมายดังนี้

ข้อมูลภาพซึ่งมีค่าตั้งแต่ 0 ถึง 255 เป็นค่าที่ได้มาจากผลรวมของค่าประจำบิตในแต่ละบิตที่ไม่เท่ากับ 0 ของข้อมูลที่มีขนาด 1 ไบต์ (8 บิต) ดังตัวอย่างต่อไปนี้

	บิตที่	8	7	6	5	4	3	2	1	
	ค่าประจำบิต	128	64	32	16	8	4	2	1	
(1)	**ข้อมูลแต่ละบิต**	1	1	1	1	1	1	1	1	
	ข้อมูลภาพคือ	128 +	64 +	32 +	16 +	8 +	4 +	2 +	1	= 255
(2)	**ข้อมูลแต่ละบิต**	1	0	0	0	0	0	0	1	
	ข้อมูลภาพคือ	128 +	0 +	0 +	0 +	0 +	0 +	0 +	1	= 129
(3)	**ข้อมูลแต่ละบิต**	0	0	0	1	1	0	0	0	
	ข้อมูลภาพคือ	0 +	0 +	0 +	16 +	8 +	0 +	0 +	0	= 24

เทอร์โบปาสคาลจะนำข้อมูลภาพจากไฟล์ มาแสดงเป็นภาพที่จอภาพด้วยหลักการต่อไปนี้คือ เทอร์โบปาสคาลจะเปลี่ยนข้อมูลภาพแต่ละไบต์ให้เป็นบิต แล้วแสดงแต่ละบิตเรียงต่อกันในแนวนอน จากซ้ายไปขวา โดยกำหนดให้บิตที่ไม่เป็น 0 เป็นจุดสว่างและบิตที่เป็น 0 ไม่สว่าง(หรือเป็นสีพื้น) ดังนั้น ข้อมูล 1 ไบต์ จึงมี 8 จุดดังตัวอย่างต่อไปนี้

ข้อมูล	ค่าแต่ละบิต	ลักษณะจุดที่แสดง
24	00011000	**
8	00001000	*
8	00001000	*
8	00001000	*
8	00001000	*
28	00011100	***

ตามตัวอย่างนี้ มีข้อมูล 6 ไบต์ ใช้แสดงภาพที่มีความกว้าง 8 พิกเซล มีความสูง 6 พิกเซล ดังนั้น ในแต่ละแถวจึงใช้ข้อมูล 1 ไบต์ และมี 6 แถว ซึ่งข้อมูลในไฟล์ไบต์ที่ 1 ถึง 4 จะเป็นดังนี้ 7 0 5 0

สำหรับภาพที่มีแถวละหลายไบต์ จะมีการแสดงข้อมูลแต่ละไบต์ด้วยวิธีการดังกล่าวนี้เรียงต่อกันไปตามลำดับ เมื่อครบตามจำนวนแล้วจึงเริ่มแสดงที่แถวล่างถัดลงมาจนครบทุกแถว ซึ่งจะทำให้เกิดภาพขึ้น 1 ภาพ

3.4.4 การเปลี่ยนไฟล์ข้อมูลภาพชนิดตัวอักษร ให้เป็นไฟล์ข้อมูลภาพแบบเทอร์โบปาสคาล ถึงแม้ว่าการสร้างภาพด้วยเอดิเตอร์เพื่อนำมาแสดงด้วยเทอร์โบปาสคาลจะสามารถทำได้ง่ายและสะดวกก็จริง แต่ยังมีข้อควรพิจารณาอยู่อีก 2 ประการคือ

ประการแรก ข้อมูลภาพที่บันทึกไว้ในไฟล์เป็นข้อมูลชนิดตัวอักษร ซึ่งจะต้องใช้เนื้อที่ในไฟล์เท่ากับหรือมากกว่าจำนวนตัวอักษรที่ใช้สร้างภาพนั้น ตามตัวอย่างที่ผ่านมาเราจะเห็นว่าไฟล์ BIRD.TXT PUPPY.TXT และ ISLAND.TXT ต่างก็มีขนาดไฟล์

เป็น 4680 ไบต์ เมื่อเปรียบเทียบกับไฟล์ BIRD.GTP ซึ่งข้อมูลเป็นรูปแบบของเทอร์โบปาสคาล ในจอภาพระบบเฮอร์คิวลิส จะมีขนาดไฟล์เพียง 640 ไบต์เท่านั้น ตามปกติขนาดของไฟล์จะมากขึ้นตามขนาดของภาพเสมอ ดังนั้น ในการบันทึกภาพขนาดเต็มจอภาพ สมมติว่าเป็นจอภาพระบบเฮอร์คิวลีส ซึ่งมีขนาด 720 x 348 เราจะต้องใช้ไฟล์ขนาดไม่ต่ำกว่า 250560 ไบต์ ซึ่งนับว่าเป็นไฟล์ที่มีขนาดใหญ่มากทีเดียว เมื่อเปรียบเทียบกับไฟล์ภาพตามรูปแบบของเทอร์โบปาสคาล ซึ่งบันทึกภาพขนาดเดียวกันนี้ด้วยไฟล์ขนาด 31360 ไบต์เท่านั้น เพราะฉะนั้น ถ้าสามารถบันทึกภาพเป็นรูปแบบของเทอร์โบปาสคาล แทนการบันทึกเป็นตัวอักษรได้ ก็จะสามารถประหยัดเนื้อที่ในดิสก์ได้

ประการที่สอง เนื่องจากโพรซีเยอร์ PutImage สามารถแสดงภาพจากไฟล์ข้อมูลภาพที่เป็นรูปแบบของเทอร์โบปาสคาลได้รวดเร็วมาก โปรแกรมก็มีขนาดเล็กและยังสามารถใช้แสดงภาพจากไฟล์ของเทอร์โบปาสคาลได้ทุกขนาด ดังนั้นถ้าเราสามารถเปลี่ยนไฟล์ภาพที่เก็บข้อมูลชนิดตัวอักษร ให้เป็นไฟล์ของเทอร์โบปาสคาลได้ การแสดงภาพจากไฟล์ก็จะสามารถทำได้อย่างสะดวกและรวดเร็ว

การเปลี่ยนไฟล์ข้อมูลภาพชนิดตัวอักษร ให้เป็นไฟล์ข้อมูลภาพแบบเทอร์โบปาสคาล สมมติว่าเปลี่ยนไฟล์ BIRD.TXT ให้เป็นไฟล์ BIRD.GTP จะมีขั้นตอนดังนี้

1. สร้างไฟล์ BIRD.GTP ให้เป็นไฟล์ชนิด TEXT เช่นเดียวกับไฟล์ BIRD.TXT แล้วบันทึกค่าความกว้างของภาพ (จำนวนอักษรใน 1 แถว - 1 = 87) ไว้ที่ไบต์ 1 และ 2 และบันทึกค่าความสูงของภาพ (จำนวนแถวทั้งหมด - 1 = 51) ไว้ที่ไบต์ที่ 3 และ 4 ตามหลักการในข้อ 3.4.3

2. อ่านข้อมูลจากไฟล์ BIRD.TXT ครั้งละ 1 แถว แล้วเปลี่ยนข้อมูลตัวอักษรแต่ละ 8 ไบต์ เรียงลำดับจากซ้ายไปขวา ให้เป็นรหัส ASCII 1 ไบต์ และบันทึกไว้ในไฟล์ BIRD.GTP โดยเรียงลำดับจากซ้ายไปขวาเช่นเดียวกัน สำหรับข้อมูลจากแถวต่อ ๆ ไปก็ดำเนินการด้วยวิธีเดียวกันนี้จนจบไฟล์ ก่อนปิดไฟล์ให้บันทึกรหัสจบภาพไว้ 1 ไบต์ ซึ่งในตัวอย่างนี้ใช้ N เป็นรหัสจบภาพ

วิธีการเปลี่ยนข้อมูลตัวอักษรจาก 8 ตัวอักษรหรือ 8 ไบต์ให้เป็นข้อมูล ASCII 1 ไบต์มีเงื่อนไขว่า ตัวอักษรแต่ละตัวคือข้อมูล 1 บิต โดยกำหนดให้อักษรตัวแรกซึ่งอยู่ทางซ้ายสุดเป็นบิตที่ 8 และเรียงตามลำดับไปทางขวาเป็น 7 6 5 4 3 2 และ1 ถ้าอักษร ณ บิตใดไม่เป็นอักษรว่าง (เครื่องหมายวรรค-spacebar) ให้นำค่าประจำบิตของอักษรเหล่านั้นมารวมกันซึ่งจะเป็นค่า ASCII ที่ต้องการ ดังตัวอย่างต่อไปนี้

บิตที่	8	7	6	5	4	3	2	1	
ค่าประจำบิต	128	64	32	16	8	4	2	1	
(1) ข้อมูลตัวอักษร	*	*	*	*	*	*	*	*	
ข้อมูล ASCII	128	64	32	16	8	4	2	1	= 255
(2) ข้อมูลตัวอักษร			*		*		*	*	
ข้อมูล ASCII	0	0	32	0	8	0	2	1	= 43
(3) ข้อมูลตัวอักษร	*							*	
ข้อมูล ASCII	128	0	0	0	0	0	0	1	= 129

ตัวอย่าง โปรแกรม 3.10 เป็นโปรแกรมทำหน้าที่เปลี่ยนไฟล์ข้อมูลภาพชนิดตัวอักษรให้เป็นไฟล์ข้อมูลภาพตามแบบของเทอร์โบปาสคาล ตามตัวอย่างนี้ โปรแกรมจะใช้ได้กับไฟล์อักษรที่มีอักษรแถวละ 88 ตัวอักษร ถ้าต้องการใช้กับไฟล์ขนาดอื่นก็ทำได้โดยเปลี่ยนค่า STRING[88] ให้เป็นค่าที่ต้องการ โปรดสังเกตว่าโปรแกรมนี้ไม่ใช้ยูนิต GRAPH เพราะไม่ใช่โปรแกรมกราฟิกส์

โปรแกรม 3.10

```
PROGRAM TextToGtpFile;
{read text file save to .gtp picture file}
USES Crt;
VAR Ch : Char;
    TextLine : STRING[88];
```

```
    PicFile   : Text;
    TextFile  : Text;

PROCEDURE SaveTextToGtp;
VAR Col,Count     : Integer;
    PicCode       : Byte;
BEGIN
  Count := 1;
  REPEAT
     PicCode := 0;              {blank}
     FOR Col := Count TO Count+7 DO
     BEGIN
        IF TextLine[Col] <> ' ' THEN
        CASE Col-(Count-1) OF
           1 : PicCode := PicCode + 128;
           2 : PicCode := PicCode +  64;
           3 : PicCode := PicCode +  32;
           4 : PicCode := PicCode +  16;
           5 : PicCode := PicCode +   8;
           6 : PicCode := PicCode +   4;
           7 : PicCode := PicCode +   2;
           8 : PicCode := PicCode +   1;
        END;
     END;
     Write(PicFile,Chr(PicCode)); {write 1 Byte}
     Inc(Count,8);
  UNTIL Count > Length(TextLine);
END;
```

```
BEGIN
  Assign(TextFile,'BIRD.TXT');
  Reset(TextFile);
  Assign(PicFile,'BIRD.GTP');
  ReWrite(PicFile);            {write headding 4 Byte}
  Write(PicFile,Chr(87),Chr(0),Chr(51),Chr(0));
  WHILE NOT Eof(TextFile) DO
  BEGIN
    ReadLn(TextFile,TextLine);
    WriteLn(TextLine);
    SaveTextToGtp;
  END;
  Write(PicFile,'N'); {write end of picture 1 Byte}
  Close(TextFile);
  Close(PicFile);
END.
```

หลังจากรันโปรแกรม 3.10 แล้ว เราจะเห็นว่าไฟล์ BIRD.GTP ที่ได้จากโปรแกรมนี้มีขนาดเพียง 577 ไบต์ ซึ่งเป็นขนาดที่เล็กกว่าไฟล์ที่ได้จากการบันทึกภาพจากจอภาพของเทอร์โบปาสคาลอีกด้วย ค่า 577 ไบต์มีที่มาดังนี้

ความกว้างและความสูงของข้อมูล 4 ไบต์

รหัสจบภาพ 1 ไบต์

ข้อมูลภาพแถวละ 88/8 = 11 ไบต์ จำนวน 52 แถว

รวมเป็น 11 x 52 = 572 ไบต์

รวมเป็นข้อมูลทั้งสิ้น 4 + 1 + 572 = 577 ไบต์

การแสดงภาพกราฟิกส์จากไฟล์ชนิด GTP ที่ได้จากโปรแกรม 3.10 ทุกไฟล์และทุกขนาดภาพ สามารถแสดงด้วยโปรแกรม 3.8 ได้ทันทีโดยไม่ต้องเปลี่ยนแปลงใดๆ ในโปรแกรม 3.8

3.4.5 การเปลี่ยนไฟล์ข้อมูลภาพจากโปรแกรมอื่น ให้เป็นไฟล์ข้อมูลภาพแบบเทอร์โบปาสคาล โปรแกรมสำเร็จรูปกราฟิกส์ส่วนใหญ่มักจะมีไฟล์ข้อมูลภาพมาให้ด้วย ข้อมูลภาพของบางโปรแกรมมีรูปแบบคล้ายกับข้อมูลภาพของเทอร์โบปาสคาลมาก เพียงแต่เราปรับปรุงข้อมูลภาพจากไฟล์เหล่านั้นบางส่วน ก็สามารถนำภาพที่มีอยู่มาใช้ได้โดยไม่ต้องเสียเวลาสร้างขึ้นใหม่ทั้งหมด เนื่องจากวิธีการเปลี่ยนข้อมูลภาพตามความคิดนี้มีส่วนเกี่ยวข้องกับความรู้พื้นฐานบางประการ ดังนั้น เพื่อความเหมาะสมจึงขอกล่าวรายละเอียดเรื่องนี้ในบทที่ 6

4

การให้สีและลายภาพ

ระบบแสดงกราฟิกส์ในเครื่องคอมพิวเตอร์แบ่งได้เป็น 2 ระบบคือ ระบบกราฟิกส์สีเดียวหรือระบบจอภาพโมโนโครม (monochrome) และระบบกราฟิกส์หลายสีหรือระบบจอภาพสี (color) การแสดงกราฟิกส์ในจอภาพโมโนโครมจะมีลักษณะเป็นจุดสว่างสีเดียว ส่วนในจอภาพสีจะแสดงกราฟิกส์ในลักษณะเป็นจุดสว่างที่มีสีแตกต่างกันหลายสี โดยกำหนดให้แต่ละจุดหรือแต่ละพิกเซลแสดงสีได้ครั้งละ 1 สี

โพรซีเยอร์และฟังก์ชันทางด้านกราฟิกส์ในเทอร์โบปาสคาล สามารถดำเนินการเกี่ยวกับสีและลายได้หลายลักษณะ และหลายขนาดตั้งแต่ขนาดเล็ก เป็นจุด เป็นเส้น เป็นรูปทรงต่างๆ จนถึงขนาดใหญ่เต็มจอภาพ จากหลายบทที่ผ่านมา เราได้คุ้นเคยกับการให้สีและลายแก่ภาพขนาดเล็กมาแล้ว ในบทนี้จะกล่าวถึงการให้สีและลายแก่ภาพที่มีบริเวณกว้าง อย่างไรก็ดี เพื่อความสะดวกแก่การศึกษา ในตอนต้นของบทจะได้กล่าวถึงการให้สีและลายแก่ภาพขนาดเล็กเพื่อเป็นการทบทวนด้วย

4.1 การให้สีแก่จุด

การให้สีในบริเวณที่เล็กที่สุดบนจอภาพได้แก่ การให้สีแก่จุด ซึ่งหมายถึงการสั่งให้คอมพิวเตอร์แสดงจุดเป็นสีใดสีหนึ่งตามที่กำหนด โพรซีเยอร์ที่ใช้คือ PutPixel

ตัวอย่าง โปรแกรม 4.1 แสดงการให้สีแก่จุดซึ่งเรียงกันเป็นแถว แถวละ 1 สีตั้งแต่สีลำดับที่ 0 ถึงลำดับที่ 15 (ดูข้อ 4.2.1) จำนวนเส้นและสีที่แสดงบนจอภาพจะแตกต่างกันตามชนิดของระบบกราฟิกส์ที่มีอยู่ในคอมพิวเตอร์ เช่น ระบบเฮอร์คิวลีส จะมี 2 เส้น จอภาพสี วีจีเอ และอีจีเอจะมี 15 เส้น โปรแกรมจะตรวจสอบจำนวนสีสูงสุดที่มีด้วยฟังก์ชัน GetMaxColor และแสดงสีตามจำนวนที่ตรวจสอบได้

โปรแกรม 4.1

```
PROGRAM PixelColor;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
     Color,X,Y        : Integer;
     MsgColor         : STRING[4];
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Y := 3;
   FOR Color := 0 TO GetMaxColor DO
   BEGIN
      Str(Color,MsgColor);
      OutTextXY(0,Y-3,'COLOR = ' + MsgColor);
      FOR X := 100 TO GetMaxX DO PutPixel(X, Y, Color);
      Inc(Y,9);
   END;
   ReadLn;
   CloseGraph
END.
```

4.2 การให้สีและลายแก่เส้นตรง

ไพรซีเยอร์ที่ใช้สำหรับแสดงเส้นตรงมี 3 ไพรซีเยอร์คือ Line LineRel และ LineTo ไพรซีเยอร์เหล่านี้จะแสดงเส้นตรงที่จอภาพด้วยสีและลายต่างๆ ซึ่งมีวิธีการกำหนดดังนี้

4.2.1 การกำหนดสี กำหนดด้วยไพรซีเยอร์ SetColor ค่าของสีที่กำหนดได้มี 16 ค่าคือ 0 ถึง 15 โดยแต่ละค่าหมายถึงแต่ละสี เช่น SetColor(9) หมายถึงกำหนดให้ใช้สีลำดับที่ 9 ค่าของสีได้กำหนดให้เป็นคอนสแตนต์ไว้ในยูนิต GRAPH ดังนี้

```
CONST
Black         =   0;    (ดำ)
Blue          =   1;    (น้ำเงิน)
Green         =   2;    (เขียว)
Cyan          =   3;    (เขียวน้ำทะเล)
Red           =   4;    (แดง)
Magenta       =   5;    (ม่วงแดง)
Brown         =   6;    (น้ำตาล)
LightGray     =   7;    (เทาอ่อน)
DarkGray      =   8;    (เทาเข้ม)
LightBlue     =   9;    (น้ำเงินอ่อน)
LightGreen    =  10;    (เขียวอ่อน)
LightCyan     =  11;    (เขียวน้ำทะเลอ่อน)
LightRed      =  12;    (แดงอ่อน)
LihtMagenta   =  13;    (ม่วงแดงอ่อน)
Yellow        =  14;    (เหลือง)
White         =  15;    (ขาว)
```

การเปลี่ยนแปลงลำดับที่ของสีสามารถทำได้โดยใช้โพรซีเยอร์ **SetPalette** ซึ่งมีรูปแบบดังนี้

SetPalette (ColorNum : Word; Color : ShortInt)

โดย ColorNum คือลำดับที่ของสีมีค่าตั้งแต่ 0 ถึง 15

Color คือสีจำนวน 16 สี

เช่น SetPalette (0, Red) หมายถึง กำหนดให้สีแดงเป็นสีลำดับที่ 0 ของสีชุดนี้ ในโปรแกรมเมื่อ SetColor(0) แล้วใช้โพรซีเยอร์ Line จะมีผลให้เส้นตรงเป็นสีแดง

สำหรับระบบกราฟิกส์ที่มีสีมากกว่า 16 สี เช่น ไอบีเอ็ม8514 และวีจีเอ ซึ่งมีสี 256 สีนั้น เราต้องเลือกกำหนดสีให้เป็นชุด โดยชุดหนึ่งมี 16 สีสำหรับวีจีเอ และมี 256 สีคือ ตั้งแต่ 0 ถึง 255 สำหรับไอบีเอ็ม8514 การกำหนดชุดของสีดังกล่าวนี้ทำได้ด้วยโพรซีเยอร์ **SetRGBPalette** รายละเอียดการใช้โพรซีเยอร์นี้ดูได้จากหนังสือ Tubor Pascal Version 6.0 Library Reference หน้า 139-140 อย่างไรก็ดี เทอร์โบปาสคาลได้อำนวยความสะดวกเกี่ยวกับการใช้สีจากระบบกราฟิกส์ที่แตกต่างกันไว้แล้ว โดยกำหนดสีลำดับที่ 0 ถึง 15 ให้เป็นสีเดียวกันในทุกระบบกราฟิกส์ที่เทอร์โบปาสคาลรู้จัก เพราะฉะนั้น เมื่อนำโปรแกรมที่สร้างด้วยระบบกราฟิกส์ชนิดหนึ่งไปใช้กับระบบกราฟิกส์ชนิดอื่น จึงสามารถใช้กันได้โดยสีไม่เปลี่ยนแปลง

ตัวอย่าง โปรแกรม 4.2 แสดงเส้นตรงสีต่างๆ ที่ระบบกราฟิกส์ในคอมพิวเตอร์เครื่องนั้นสามารถแสดงได้ จำนวนสีจะมีค่าตั้งแต่ 2 ถึง 15 โดย 2 หมายถึงระบบโมโนโครม

โปรแกรม 4.2

```
PROGRAM LineColor;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
     Color,X,Y         : Integer;
     MsgColor          : STRING[4];
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   X := 100; Y := 3;
   FOR Color := 0 TO GetMaxColor DO
   BEGIN
      Str(Color,MsgColor);
      OutTextXY(0,Y-3,'COLOR = ' + MsgColor);
      SetColor(Color);
      Line(X, Y, GetMaxX, Y);
      Inc(Y,9);
   END;
   ReadLn;
   CloseGraph
END.
```

4.2.2 การกำหนดลาย ความหมายของลาย (pattern) คือ เส้นตรงที่ไม่เป็นเส้นทึบ ตามปกติเมื่อกำหนดสีตามข้อ 4.2.1 แล้ว เส้นตรงที่ได้จะเป็นเส้นทึบและมีสีตามที่กำหนดด้วย SetColor ถ้าต้องการเปลี่ยนเส้นทึบให้เป็นลาย ก็สามารถทำได้ด้วยโพรซีเยอร์ SetLineStyle ซึ่งได้กำหนดลายและความหนาของเส้นตรงเป็นคอนสแตนต์ไว้ดังนี้

```
CONST
SolidLn      =   0;    (เส้นทึบ)
DottedLn     =   1;    (เส้นจุด)
```

```
CenterLn     =  2;   (เส้นขีดกลาง)
DashedLn     =  3;   (เส้นขีดล่าง)
UserBitLn    =  4;   (เส้นที่กำหนดเอง)
NormWidth    =  1;   (เส้นหนาปกติ)
ThickWidth   =  3;   (เส้นหนา 3 เท่า)
```

เทอร์ไบปาสคาลสร้างลายมาให้ 3 แบบ และเปิดโอกาสให้เราสร้างลายได้เองด้วย โดยแต่ละลายเป็นข้อมูลขนาด 16 บิต ซึ่งทำให้เราสามารถออกแบบลายให้แตกต่างกันได้เป็นจำนวน 2^{16} ลายหรือเท่ากับ 65536 ลาย การกำหนดลายขึ้นเองมีรายละเอียดอยู่ในบทที่ 6

ตัวอย่าง โปรแกรม 4.3 แสดงเส้นตรงที่มี สี ลาย และความหนาของเส้นแตกต่างกัน ในโปรแกรมนี้ ใช้เส้นทึบและลายที่มีอยู่ในเทอร์ไบปาสคาลรวม 3 ลาย และลายที่สร้างเองโดยไม่กำหนดค่า ใช้ความหนาของเส้น 2 ขนาด จำนวนสีรวม 16 สี โปรแกรมจะแสดงเส้นตรงตามลำดับของ ลาย (Styte) สี (Color) และความหนา (Thick) โดยจะมีการเปลี่ยนค่าทุกครั้งที่กดคีย์ Enter สำหรับระบบเฮอร์คิวลีสจะมีเส้นรวม 20 เส้น ส่วนจอภาพสีที่ให้สีได้ถึง 16 สีจะมีเส้นรวม 160 เส้น

โปรแกรม 4.3

```
PROGRAM LineStyleColor;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
     Color,Style,X,Y       : Integer;
     MsgStyle,MsgColor     : STRING[4];
BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
```

```
  FOR Style := 0 TO 4 DO
  BEGIN
     X := 300; Y := 3;
     FOR Color := 0 TO GetMaxColor DO
     BEGIN
        SetColor(Color);
        Str(Style,MsgStyle);
        Str(Color,MsgColor);
        OutTextXY(0,Y-3,'LINE STYLE = ' + MsgStyle);
        OutTextXY(120,Y-3,'COLOR = ' + MsgColor);
        OutTextXY(200,Y-3,'THICK = 1 ');
        SetLineStyle(Style,Color,1);
        Line(X, Y, GetMaxX, Y);
        Inc(Y,9);
        OutTextXY(200,Y-3,'THICK = 3 ');
        SetLineStyle(Style,Color,3);
        Line(X, Y, GetMaxX, Y);
        Inc(Y,9);
     END;
     ReadLn;
     ClearDevice;
  END;
  CloseGraph
END.
```

4.3 การให้สีและลายแก่ภาพสี่เหลี่ยม

เทอร์โบปาสคาลมีโพรซีเยอร์สำหรับใช้แสดงภาพสี่เหลี่ยมขนาดต่างๆ ตามกำหนดรวม 2 โพรซีเยอร์คือ **Rectangle** และ **Bar** ความแตกต่างของโพรซีเยอร์ทั้ง

สองนี้คือ Rectangle จะแสดงภาพสี่เหลี่ยมโดยมีสี และลายของเส้นภาพตามที่กำหนดไว้ด้วยโพรซีเยอร์ SetColor และ SetLineStyle ส่วน Bar จะแสดงภาพสี่เหลี่ยมโดยมีสีและลายอยู่เต็มภายในภาพ สีและลายกำหนดโดยโพรซีเยอร์ **SetFillStyle** ซึ่งมีรูปแบบดังนี้

SetFillStyle (Pattern : Word; Color: Word)

โดย Color คือสีมีค่า 0 ถึง 15

Pattern คือลายซึ่งเทอร์โบปาสคาลได้กำหนดเป็นคอนสแตนต์ไว้ดังนี้

```
CONST
EmptyFill          =   0;   (สีพื้นไม่มีลาย)
SolidFill          =   1;   (สีที่กำหนดไม่มีลาย)
LineFill           =   2;   (ลายเส้น --- )
LtSlashFill        =   3;   (ลาย /// )
SlashFill          =   4;   (ลาย /// เส้นหนา)
BkSlashFill        =   5;   (ลาย \\\ เส้นหนา)
LtBkSlashFill      =   6;   (ลาย \\\ )
HatchFill          =   7;   (ลายตารางสี่เหลี่ยมมุมฉาก)
XHatchFill         =   8;   (ลายตารางสี่เหลี่ยมขนมเปียกปูน)
InterleaveFill     =   9;   (ลายตาข่ายละเอียด)
WideDotFill        =  10;   (ลายจุดห่างกัน)
CloseDotFill       =  11;   (ลายจุดชิดกัน)
UserFill           =  12;   (ลายกำหนดเอง)
```

ลายที่กำหนดไว้แล้วมี 10 ลายคือ ตั้งแต่เลข 2 ถึง 11 สำหรับการกำหนดลายเองสามารถทำได้ด้วยโพรซีเยอร์ **SetFillPattern** ซึ่งเป็นการกำหนดลายด้วยข้อมูลขนาด 8 ไบต์ รายละเอียดของโพรซีเยอร์นี้ดูได้จากหนังสือ Turbo Pascal Version 6.0 Library Reference หน้า 130-131

ตัวอย่าง โปรแกรม 4.4 แสดงการให้สีและลายแก่ภาพสี่เหลี่ยม ซึ่งจะแสดงให้เห็นในลักษณะของตารางที่ประกอบด้วยสี 16 สี และลาย 13 ลาย จำนวนของภาพสี่เหลี่ยมที่แสดง ขึ้นอยู่กับจำนวนสีที่ระบบกราฟิกส์นั้นมี เช่น ระบบเฮอร์คิวลีส จะมีภาพเพียง 26 ภาพ เนื่องจากมี 2 สี

โปรแกรม 4.4

```
PROGRAM BarStyleColor;
USES Graph;
VAR  GraphDriver,GraphMode     : Integer;
     Color,Style,X1,Y1,StartX : Integer;
     MsgStyle,MsgColor        : STRING[4];
BEGIN
     GraphDriver := Detect;
     InitGraph(GraphDriver,GraphMode,'');

     FOR Color := 0 TO GetMaxColor DO     {column heading}
     BEGIN
        Str(Color,MsgColor);
        OutTextXY((Color+1)*38,0,'CLR' + MsgColor);
     END;

     X1 := 42; Y1 := 15; StartX := X1;   {set position of bar}
     FOR Style := 0 TO 12 DO
     BEGIN
        FOR Color := 0 TO GetMaxColor DO
        BEGIN
           Str(Style,MsgStyle);       {column heading}
           OutTextXY(0,Y1+7,'STL' + MsgStyle);
```

```
        SetFillStyle(Style,Color);
        Bar(X1, Y1, X1+30, Y1+20);
        Inc(X1,37);
      END;
      Inc(Y1,25); X1 := StartX;
    END;
    ReadLn;
    CloseGraph
END.
```

ตัวอย่าง โปรแกรม 4.5 แสดงการให้สีและลายขนาดเต็มจอภาพ ขอให้ทดลองเปลี่ยนค่าลาย (0 – 11) และ ค่าสี (0 – 15) ในโพรซีเยอร์ SetFillStyle เพื่อเปรียบเทียบภาพที่เห็น

โปรแกรม 4.5

```
PROGRAM BarFillScreen;
USES Graph;
VAR  GraphDriver,GraphMode    : Integer;
   Color,Style : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Style := 8;  Color := 4;
   SetFillStyle(Style,Color);
   Bar(0, 0, GetMaxX, GetMaxY);
   ReadLn;
   CloseGraph
END.
```

4.4 การให้สีและลายแก่ภาพสี่เหลี่ยมแบบ 3 มิติ

ไพรซีเยอร์แสดงภาพสี่เหลี่ยมแบบ 3 มิติคือ **Bar3D** ซึ่งมีรูปแบบดังนี้

Bar3D(X1,Y1, X2,Y2 :Integer; Depth :Word; Top :Boolean)

โดย X1,Y1 คือมุมบนซ้ายของภาพสี่เหลี่ยม

X2,Y2 คือมุมล่างขวาของภาพสี่เหลี่ยม

Depth คือความลึกของภาพด้านข้างทางขวามือ

Top คือความลึกของภาพด้านบนซึ่งจะแสดงให้เห็นถ้า Top เป็น True และไม่แสดงถ้า Top เป็น False

ไพรซีเยอร์ Bar3D จะแสดงสีและลายในภาพสี่เหลี่ยม โดยใช้สีและลายที่กำหนดด้วยไพรซีเยอร์ SetFillStyle หรือ SetFillPattern ส่วนเส้นแสดงความลึกจะมีสีและลายตามที่กำหนดด้วย SetLineStyle และ SetColor ค่าความลึกของภาพสามารถกำหนดได้ตามต้องการ ส่วนใหญ่นิยมกำหนดให้มีค่าเป็น 25% ของความกว้าง ซึ่งเขียนเป็นสเตตเมนต์ได้ดังนี้

Bar3D (X1,Y1, X2,Y2, (X2 - X1 + 1) DIV 4, TopOn)

ตัวอย่าง โปรแกรม 4.6 แสดงภาพสี่เหลี่ยมแบบ 3 มิติซึ่ง**มีภาพ**สี่เหลี่ยม 2 ภาพซ้อนกัน

โปรแกรม 4.6

```
PROGRAM Bar3Dimension;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     X1,Y1,X2,Y2   : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
```

```
    X1 := 10; Y1 := 300; X2 := 300;  Y2 := 150;
    SetFillStyle(CloseDotFill,Yellow);
    SetColor(Blue);
    Bar3D(X1, Y1, X2, Y2, (X2-X1+1) DIV 4, TopOn); {large bar3d}
    SetColor(Cyan);
    SetFillStyle(XHatchFill,LightRed);
    X1 := 100; Y1 := 135; X2 := 200; Y2 := 50;
    Bar3D(X1, Y1, X2, Y2, (X2-X1+1) DIV 5, TopOn); {small bar3d}
    ReadLn;
    CloseGraph
END.
```

4.5 การให้สีและลายแก่ภาพหลายเหลี่ยม

การแสดงภาพหลายเหลี่ยมที่มีสีและลายอยู่ภายในทำได้ด้วยโพรซีเยอร์ **FillPoly** ซึ่งมีรูปแบบดังนี้

```
FillPoly (NumPoints : Word; VAR PolyPoints)
```

โดย PolyPoints คือค่า X,Y ของแต่ละด้านของภาพหลายเหลี่ยม

NumPoints คือจำนวนจุดโคออร์ดิเนตของภาพหลายเหลี่ยม

ภาพหลายเหลี่ยมที่สร้างด้วยโพรซีเยอร์ Fillpoly จะมีสีและลายอยู่ภายในตามที่กำหนดด้วยโพรซีเยอร์ SetFillStyle หรือ FillPattern ส่วนเส้นรอบภาพจะมีสีและลายตามที่กำหนดด้วยโพรซีเยอร์ SetLineStyle

---

ตัวอย่าง โปรแกรม 4.7 แสดงภาพหลายเหลี่ยมซึ่งมีสีและลายอยู่ภายใน

โปรแกรม 4.7

```
PROGRAM FilPolygon;
USES Crt,Graph;
CONST House : ARRAY[1..6] OF PointType = ((X : 100; Y : 300),
                                          (X : 600; Y : 300),
                                          (X : 600; Y : 250),
                                          (X : 350; Y :  50),
                                          (X : 100; Y : 250),
                                          (X : 100; Y : 300));
VAR  GraphDriver,GraphMode     : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   SetLineStyle(DottedLn,0,Green);
   SetFillStyle(InterleaveFill,Magenta);
   FillPoly(6, House);
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

4.6 การให้สีและลายแก่ภาพวงรี

การแสดงภาพวงรีที่มีสีและลายอยู่ภายในทำได้ด้วยโพรซีเยอร์ FillEllipse ซึ่งมีรูปแบบดังนี้

FillEllipse (X, Y : Integer; XRadius, YRadius : Word)

โดย X,Y คือจุดศูนย์กลางของวงรี

XRadius คือค่ารัศมีในแนวนอน

YRadius คือค่ารัศมีในแนวตั้ง

ภาพวงรีจะมีสีและลายอยู่ภายในตามที่กำหนดด้วยโพรซีเยอร์ SetFillstyle ส่วนเส้นรอบภาพจะมีสีตามที่กำหนดด้วยโพรซีเยอร์ SetColor

---

ตัวอย่าง โปรแกรม 4.8 แสดงภาพวงรีที่มีสีและลายอยู่ภายในภาพ

โปรแกรม 4.8

```
PROGRAM FillStColEllipse;
USES Graph;
VAR  GraphDriver,GraphMode        : Integer;
     XMax,YMax,XRadius,YRadius  : Integer;
     XCenter,YCenter,Count      : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   XMax    := GetMaxX;
   YMax    := GetMaxY;
   XCenter := XMax DIV 2;
   YCenter := YMax DIV 2 - 20;
   YRadius  := YCenter;
   XRadius  := XCenter DIV 10;
   SetColor(White);
   SetFillStyle(10,1);
   FillEllipse(XCenter,YCenter,XRadius,YRadius);   { midle tree }
   SetFillStyle(3,8);
   FillEllipse(XCenter + 60, YCenter + 30,           { right trees }
            XRadius + 20, YRadius - 30);
   SetFillStyle(4,3);
```

```
    FillEllipse(XCenter + 130, YCenter + 70,
                XRadius + 30, YRadius - 70);
    SetFillStyle(5,4);
    FillEllipse(XCenter + 200, YCenter + 100,
                XRadius + 30, YRadius - 100);
    SetFillStyle(8,5);
    FillEllipse(XCenter - 60, YCenter + 30,          { left trees }
                XRadius + 20, YRadius - 30);
    SetFillStyle(6,6);
    FillEllipse(XCenter - 130, YCenter + 70,
                XRadius + 30, YRadius - 70);
    SetFillStyle(2,7);
    FillEllipse(XCenter - 200, YCenter + 100,
                XRadius + 30, YRadius - 100);
    SetColor(2);
    Line(XCenter - 200, YMax - 40, XCenter - 200, YMax);
    Line(XCenter - 130, YMax - 40, XCenter - 130, YMax);
    Line(XCenter - 60,  YMax - 40, XCenter - 60,  YMax);
    Line(XCenter      , YMax - 40, XCenter      , YMax);
    Line(XCenter + 200, YMax - 40, XCenter + 200, YMax);
    Line(XCenter + 130, YMax - 40, XCenter + 130, YMax);
    Line(XCenter + 60,  YMax - 40, XCenter + 60,  YMax);
    SetColor(14);
    Line(50, YMax, XMax - 50, YMax);   { ground }
    ReadLn;
    CloseGraph
END.
```

4.7 การให้สีและลายแก่ภาพพาย

ภาพวงกลมแบบกราฟพาย (pie) ที่มีสีและลายอยู่ภายในสามารถแสดงได้ด้วยโพรซีเยอร์ PieSlice ซึ่งมีรูปแบบดังนี้

PieSlice (X,Y :Integer; StAngle, EndAngle, Radius :Word)

โดย X,Y คือจุดศูนย์กลางของวงกลม

Radius คือรัศมีวงกลม

StAngle คือมุมเริ่มต้นแสดงภาพ

EndAngle คือมุมหยุดแสดงภาพ

ภาพวงกลมพายจะเริ่มแสดงตั้งแต่ StAngle จนถึง EndAngle ถ้ากำหนดให้ StAngle เป็น 0 และ EndAngle เป็น 360 จะได้ภาพเป็นวงกลม ค่าของมุมมีหน่วยเป็นองศา เริ่มจาก 0 องศาที่ตำแหน่ง 3 นาฬิกา แล้วนับเพิ่มขึ้นในทิศทางทวนเข็มนาฬิกาคือเป็น 90 องศาที่ 12 นาฬิกา เป็น 180 องศาที่ 9 นาฬิกา วนเช่นนี้ไปตามลำดับ สีและลายในภาพวงกลมแบบพายจะเป็นตามที่กำหนดด้วยโพรซีเยอร์ SetFillStyle หรือ SetFillPattern ส่วนเส้นขอบภาพจะมีสีตามที่กำหนดด้วยโพรซีเยอร์ SetColor

---

ตัวอย่าง โปรแกรม 4.9 แสดงการให้สีและลายแก่ภาพวงกลมแบบพาย ซึ่งในที่นี้เป็นภาพไม่เต็มวงกลม

โปรแกรม 4.9

```
PROGRAM FillPieSlice;
USES Crt,Graph;
VAR  GraphDriver,GraphMode : Integer;
     XMax,YMax,YRadius  : Integer;
     XCenter,YCenter    : Integer;
```

```
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   XMax    := GetMaxX;
   YMax    := GetMaxY;
   XCenter := XMax DIV 2;
   YCenter := YMax DIV 2 - 20;
   YRadius  := YCenter;
   SetColor(White);
   SetFillStyle(LtSlashFill,LightGreen);
   PieSlice(XCenter,YCenter,0,340,YRadius);
   ReadLn;
   CloseGraph
END.
```

4.8 การให้สีและลายที่บริเวณใด ๆ บนจอภาพ

สำหรับการให้สีและลาย ณ บริเวณใดๆ บนจอภาพ สามารถทำได้โดยใช้โพรซีเยอร์ **FloodFill** ซึ่งมีรูปแบบดังนี้

FloodFill (X, Y : Integer; Border : Word)

โดย X,Y คือตำแหน่งเริ่มต้นของการให้สีและลาย

Border คือเส้นขอบภาพซึ่งเป็นสีใดสีหนึ่ง

โพรซีเยอร์ FloodFill จะให้สีและลาย โดยเริ่มต้นจาก X,Y ที่กำหนดและไปสิ้นสุดที่ Border ซึ่งเป็นขอบของภาพ ถ้า X,Y อยู่ภายใน Border สีและลายจะอยู่ในภาพ แต่ถ้า X,Y อยู่นอกภาพ หรือ ถ้ามีช่องว่างที่ Border สีและลายจะออกไปนอกบริเวณที่กำหนดทางช่องว่างนั้น

สีและลายกำหนดด้วยโพรซีเยอร์ SetFillStyle หรือ SetFillPattern สีของภาพกำหนดด้วยโพรซีเยอร์ SetColor

---

ตัวอย่าง โปรแกรม 4.10 แสดงการให้สีและลายที่จอภาพว่าง ซึ่งมีเฉพาะสีพื้นสี และลายจะปรากฏเต็มจอภาพ โดยจะมีการเปลี่ยนแปลงทุกครั้งที่กดแป้น Enter ระบบกราฟิกส์เฮอร์คิวลีสจะแสดงได้ 24 ภาพ ส่วนระบบกราฟิกส์ 16 สีจะแสดงได้ 192 ภาพ

โปรแกรม 4.10

```
PROGRAM ScreenStyleColor;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;
     Color,Style           : Integer;
     MsgStyle,MsgColor     : STRING[4];
BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  FOR Style := 0 TO 11 DO
  FOR Color := 0 TO GetMaxColor DO
  BEGIN
    ClearDevice;
    SetColor(Color);
    Str(Style,MsgStyle);
    Str(Color,MsgColor);
    OutTextXY(0,0,'STYLE = ' + MsgStyle);
    OutTextXY(100,0,'COLOR = ' + MsgColor);
    SetFillStyle(Style,Color);
    FloodFill(0,9,GetColor);
```

```
      ReadLn
    END;
    CloseGraph
END.
```

ตัวอย่าง โปรแกรม 4.11 แสดงการให้สีและลายแก่ภาพชิ้นแตงโม โปรแกรมจะเริ่มด้วยการแสดงเส้นขอบภาพ เมื่อกดคีย์ Enter แต่ละครั้ง จะมีการให้สีและลายแก่ส่วนที่เป็นเนื้อแตงโม โดยจะมีการเปลี่ยนลาย 10 ครั้ง

โปรแกรม 4.11

```
PROGRAM WaterMelonC;
USES Graph;
VAR  GraphDriver,GraphMode,
      X,Y,StartAngle,EndAngle,Radius,Style : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   X := 300;  Y := 150;  Radius := 120;
   StartAngle := 180;
   EndAngle   := 360;
   FOR Style := 0 TO 10 DO
   BEGIN
      Arc(X,Y,StartAngle,EndAngle,Radius);
      Arc(X,Y,StartAngle,EndAngle,Radius-1);
      Arc(X,Y,StartAngle,EndAngle,Radius-2);
      Arc(X,Y,StartAngle,EndAngle,Radius-10);
      Line(X-Radius,Y,X+Radius,Y);
```

```
    SetFillStyle(Style,Style);
    FloodFill(X,Y+10,GetMaxColor);
    ReadLn;
    ClearViewport
  END;
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 4.12 แสดงการให้สีและลายแก่ภาพดิสเก็ตที่เก็บอยู่ในซอง โปรแกรมจะเริ่มด้วยการแสดงเส้นขอบภาพ เมื่อกดคีย์ Enter แต่ละครั้ง จะมีการให้สีทีละส่วนจนครบทุกส่วน

โปรแกรม 4.12

```
PROGRAM FillDiskette;
USES Graph;
CONST MaxVector = 22;
   Disk : ARRAY[1..MaxVector] OF STRING[7] = (
         'M100000','L500000','L500230','L100230','L100000',
         'M115230','L115330','L485330','L485245','L475245','L475230
         'M120327','L280327','L280250','L120250','L120327',
         'M100230','L100300','L115300',
         'M500230','L500300','L485300');
TYPE
  Pointer = ^Pixel;
  Pixel   = RECORD
              Visible : Boolean;
              X       : Real;
```

```
                Y         : Real;
                Next      : Pointer;
              END;
VAR
  VectorFile  : Text;
  Points      : Pointer;
  GraphDriver,GraphMode : Integer;

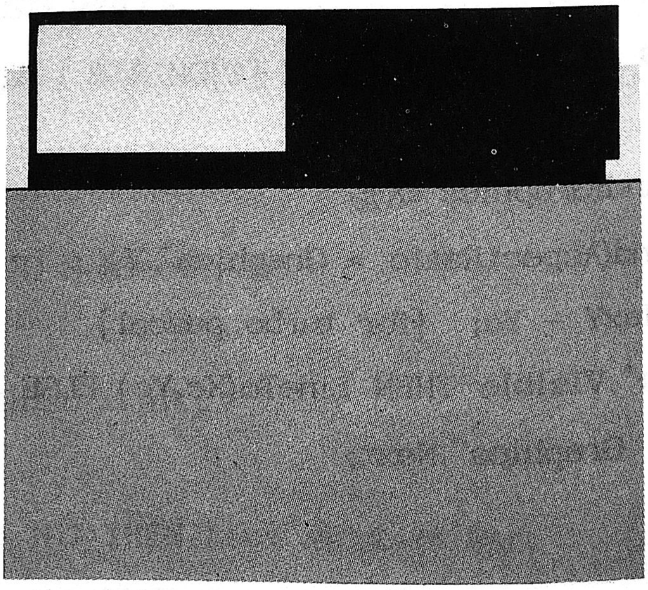
FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
  GetAspectRatio(Xasp,Yasp);
  AspectRatio := Xasp/Yasp
END;

PROCEDURE ReadArray(VAR Points : Pointer);
VAR Last,Ptr     : Pointer;
    Command      : Char;
    Code,Count   : Integer;
    OneLine      : STRING[7];
    StringX,StringY : STRING[3];
BEGIN
  Points := Nil;
  FOR Count := 1 TO MaxVector DO
  BEGIN
     OneLine := Disk[Count];
     New(Ptr);
     IF PointS <> Nil THEN Last^.Next := Ptr;
     IF PointS =  Nil THEN Points := Ptr;
```

```
    WITH Ptr^ DO
    BEGIN
       Command  := OneLine[1];
       StringX  := Copy(OneLine,2,3);
       StringY  := Copy(OneLine,5,3);
       IF Command IN['m','M'] THEN Visible := False ELSE
       IF Command IN['l','L'] THEN Visible := True;
       Val(StringX,X,Code);
       Val(StringY,Y,Code);
    END;
    Last := Ptr;
  END;
  Last^.Next := Nil;
END;

PROCEDURE DisplayDrawing(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
  Graphics := Points;
  WHILE Graphics <> Nil DO
  BEGIN
     Xc  := Round(Graphics^.X);
     Yc  := Round(AspectRatio * Graphics^.Y);
     Yc  := GetMaxY - Yc;  {for turbo pascal}
     IF Graphics^.Visible THEN LineTo(Xc,Yc) ELSE MoveTo(Xc,Yc);
     Graphics := Graphics^.Next;
  END;
END;
```

```
BEGIN
   ReadArray(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   DisplayDrawing(Points);
   ReadLn;
   SetFillStyle(1,Yellow);
   FloodFill(300,300,GetColor);  ReadLn;
   SetFillStyle(1,Blue);
   FloodFill(121,232,GetColor);  ReadLn;
   SetFillStyle(1,Green);
   FloodFill(121,180,GetColor);  ReadLn;
   SetFillStyle(9,Yellow);
   FloodFill(490,180,GetColor);  ReadLn;
   SetFillStyle(9,Yellow);
   FloodFill(103,180,GetColor);  ReadLn;
   CloseGraph
END.
```



รูปที่ 4.1 ภาพที่ได้จากโปรแกรม 4.12

4.9 การหาจุด X, Y ที่อยู่ภายในภาพ

การให้สีและลายภาพด้วยโพรซีเยอร์ FloodFill จะเป็นวิธีการที่สะดวกที่สุดสำหรับภาพทั่วๆ ไปที่เราทราบค่า X,Y ซึ่งอยู่ภายในภาพ เพราะโพรซีเยอร์นี้มีพารา-มิเตอร์น้อยกว่าโพรซีเยอร์อื่น สิ่งที่จะต้องพิจารณาเพิ่มเติมสำหรับการให้สีและลายด้วยโพรซีเยอร์ FloodFill ก็คือวิธีการหาค่า X,Y ซึ่งจะต้องอยู่ภายในขอบเขตของภาพนั้นอย่างแน่นอน วิธีการเบื้องต้นสำหรับคำนวณหาค่า X,Y จากภาพลักษณะต่างๆ มีตัวอย่างดังนี้

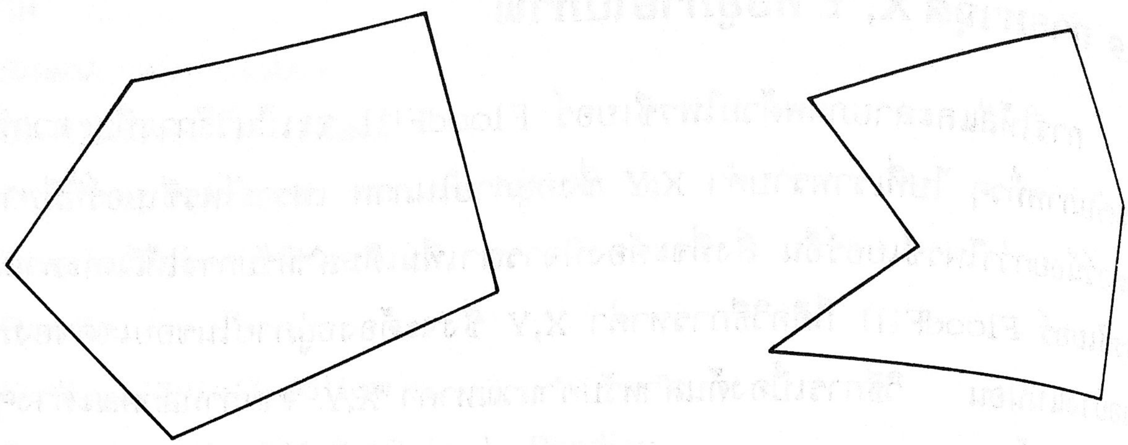
4.9.1 ภาพสี่เหลี่ยม ซึ่งมี X1,Y1 เป็นตำแหน่งมุมบนซ้ายและ X2,Y2 เป็นตำแหน่งมุมล่างขวา จะหาค่า X,Y ที่อยู่ภายในได้ดังนี้

```
X = (X1 + X2) / 2
Y = (Y1 + Y2) / 2
```

4.9.2 ภาพสี่เหลี่ยม ซึ่งมีตำแหน่งมุมทั้งสามเป็น (X1,Y1), (X2,Y2) และ (X3,Y3) จะหาค่า X,Y ที่อยู่ภายในได้ดังนี้

```
X = (X1 + X2 + X3) / 3
Y = (Y1 + Y2 + Y3) / 3
```

4.9.3 ภาพหลายเหลี่ยม ภาพหลายเหลี่ยมที่มี n ด้าน จะมีตำแหน่งมุมต่างๆ เป็น (X1,Y1), (X2,Y2),...,(Xn,Yn) ภาพหลายเหลี่ยมมี 2 แบบคือ แบบนูน (convex) และแบบเว้า (nonconvex) ดังแสดงในรูปที่ 4.2



รูปที่ 4.2 ภาพหลายเหลี่ยมแบบนูนและแบบเว้า

ค่า X, Y ที่อยู่ภายในภาพหลายเหลี่ยมแบบนูนจะหาได้ดังนี้

X = (X1 + X2 + ... + Xn) / n

Y = (Y1 + Y2 + ... + Yn) / n

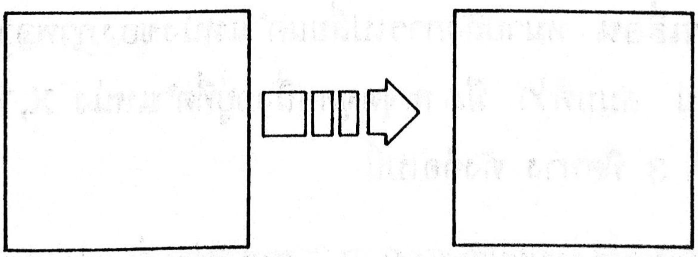
5

การแปลงภาพกราฟิกส์

การแปลง (transformation) ในความหมายทั่วไปหมายถึง การทำให้แตกต่างไปจากเดิม เพราะฉะนั้น การแปลงภาพจึงหมายถึงการทำให้ภาพแตกต่างไปจากเดิม วิธีการแปลงภาพอาจแบ่งได้เป็น 2 ประเภทคือ การแปลงภาพขั้นเดียวและการแปลงภาพหลายขั้นซึ่งมีรายละเอียดดังต่อไปนี้

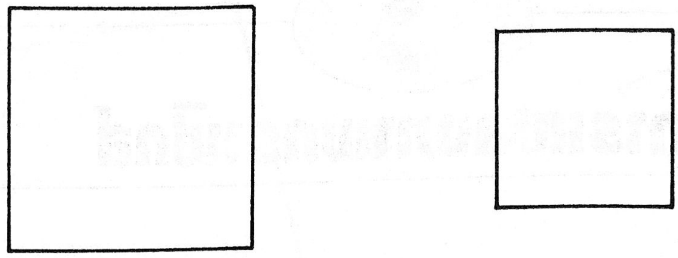
5.1 การแปลงภาพขั้นเดียว

การแปลงภาพขั้นเดียวหมายถึง การแปลงภาพที่มีเพียงขั้นตอนเดียว ซึ่งประกอบด้วยวิธีการแปลงภาพ 4 วิธีคือ การเลื่อน (translation) การกำหนดสเกล (scaling) การหมุน (rotation) และการเบ้ (shears)

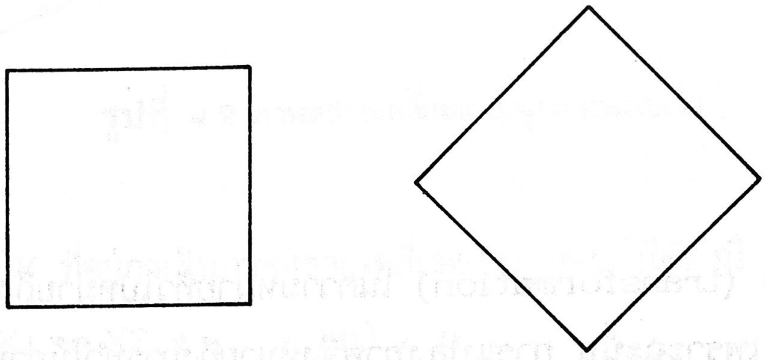


การเลื่อน

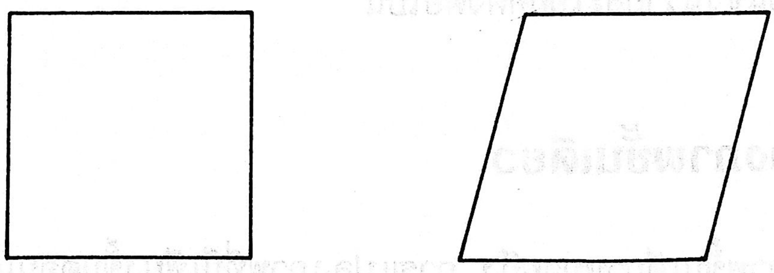
รูปที่ 5.1 ลักษณะการแปลงภาพด้วยวิธีต่าง ๆ



การกำหนดสเกล



การหมุน



การเบ้

รูปที่ 5.1 (ต่อ) ลักษณะการแปลงภาพด้วยวิธีต่าง ๆ

5.1.1 การเลื่อน หมายถึงการเปลี่ยนตำแหน่งของภาพจากตำแหน่งเดิม ให้ไปอยู่ที่ตำแหน่งใหม่ สมมติว่า มีภาพจุดจุดหนึ่งอยู่ที่ตำแหน่ง X,Y เราสามารถเลื่อนภาพจุดดังกล่าวนี้ได้ 3 ทิศทาง ดังต่อไปนี้

1. เลื่อนในทิศทางขนานกับแกน X สมมติว่าเลื่อนห่างจากตำแหน่งเดิม (X) เป็นระยะทางเท่ากับ TransX ตำแหน่ง X ใหม่จะหาได้จากสมการต่อไปนี้

X = X + TransX

2. เลื่อนในทิศทางขนานกับแกน Y สมมติว่าเลื่อนห่างจากตำแหน่งเดิม (Y) เป็นระยะทางเท่ากับ TransY ตำแหน่ง Y ใหม่จะหาได้จากสมการต่อไปนี้

Y = Y + TransY

3. เลื่อนในทิศทางใดๆ จากแกน X และแกน Y สมมติว่า เลื่อนห่างจากตำแหน่งเดิม (X,Y) เป็นระยะทาง TransX และ TransY ตามลำดับ ตำแหน่ง X,Y ใหม่ จะหาได้จากสมการต่อไปนี้

X = X + TransX

Y = Y + TransY

ตัวอย่างโพรซีเยอร์ที่ใช้สำหรับเลื่อนภาพจุดจุดหนึ่งไปในทิศทางใดๆ จากแกน X และ แกน Y แสดงดังต่อไปนี้

```
PROCEDURE Translate(Points : Pointer; TransX,TransY : Real);
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      X := X + TransX;
      Y := Y + TransY;
      Points := Next
   END;
END;
```

ข้อตกลงเกี่ยวกับโปรแกรมในบทนี้

เนื่องจากโปรแกรมตัวอย่างหลายโปรแกรมในบทนี้มีการใช้ข้อมูล ฟังก์-ชันและโพรซีเยอร์ต่างๆ ร่วมกัน ดังนั้นเพื่อหลีกเลี่ยงการอธิบายซ้ำกันหลายครั้ง จึงได้นำสิ่งที่จะปรากฏอยู่ในทุกโปรแกรมมาอธิบายรวมกันไว้ในที่นี้ ดังต่อไปนี้

1. ชนิดของข้อมูล กำหนดให้เป็นพอยน์เตอร์ดังนี้

```
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
                Visible : Boolean;
                X       : Real;
                Y       : Real;
                Next    : Pointer;
           END;
VAR
   GraphDriver,GraphMode : Integer;
   Points        : Pointer;
```

Pointer	เป็นพอยน์เตอร์ชี้ไปที่ Pixel
Pixel	เป็นเรคอร์ด ซึ่งประกอบด้วยฟีลด์ที่มีความหมายดังนี้
Visible	สำหรับกำหนดให้แสดงพิกเซลเมื่อเป็น True และไม่แสดงพิกเซลเมื่อเป็น False
X,Y	เป็นตำแหน่ง X และ Y บนจอภาพ
Next	เป็นพอยน์เตอร์ชี้เรคอร์ดต่อไป

ตัวอย่างการกำหนดค่า

```
ถ้า  Visible  :=  False;
     X        :=  100;
     Y        :=  100;
```

หมายถึงไม่ให้แสดงพิกเซลที่ตำแหน่ง 100,100 บนจอภาพ

```
ถ้า  Visible  :=  True;
     X        :=  100;
     Y        :=  100;
```

หมายถึงให้แสดงพิกเซลที่ตำแหน่ง 100, 100 บนจอภาพ

2. โพรซีเยอร์ Set Pointer ใช้สำหรับกำหนดเงื่อนไขให้แสดงภาพหรือไม่แสดงภาพ ณ จุด X, Y บนจอภาพ โพรซีเยอร์ SetPointer เป็นดังนี้

```
PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

  PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
  BEGIN
    New(Ptr);
    IF PointS <> Nil THEN Last^.Next := Ptr;
    IF PointS =  Nil THEN Points := Ptr;
    WITH Ptr^ DO
    BEGIN
      Visible := VisibleC;
      X       := Xc;
```

```
        Y       := Yc
      END;
      Last := Ptr;
    END;

BEGIN
    Points := Nil;
    SetPointer(False, 100, 100);
    SetPointer(True,  100, 100);
    Last^.Next := Nil;
END;
```

โพรซีเยอร์ SetPointer จะอำนวยความสะดวกในการกำหนดเงื่อนไขและตำแหน่งต่างๆ เพื่อการสร้างภาพบนจอภาพ ตามตัวอย่างนี้ เป็นการกำหนดให้สร้างภาพจุดหนึ่งจุด ณ ตำแหน่ง 100,100 บนจอภาพ

3. โพรซีเยอร์ Draw ใช้สำหรับแสดงภาพตามเงื่อนไขและตำแหน่งต่างๆ ตามที่กำหนดไว้ด้วยโพรซีเยอร์ SetPointer โพรซีเยอร์ Draw เป็นดังนี้

```
PROCEDURE Draw(Points : Pointer);
VAR Graphics  : Pointer;
    Xc,Yc     : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
```

```
  BEGIN
     Xc   := Round(Graphics^.X);
     Yc   := Round(AspectRatio * Graphics^.Y);
     Yc   := GetMaxY - Yc;    {TrueY for turbo pascal}
     IF Graphics^.Visible THEN LineTo(Xc,Yc)
                          ELSE MoveTo(Xc,Yc);
     Graphics := Graphics^.Next;
  END;
END;
```

---

4. *โพรซีเยอร์ Blank* ใช้สำหรับลบภาพที่แสดงไว้ครั้งล่าสุด ส่วนใหญ่จะใช้โพรซีเยอร์นี้เพื่อลบภาพเดิมก่อนที่จะเลื่อนไปตำแหน่งใหม่ โพรซีเยอร์ Blankเป็นดังนี้

```
PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
   Color := GetColor;
   SetColor(GetBkColor);
   Draw(Points);
   SetColor(Color);
END;
```

---

5. *ฟังก์ชัน Aspect Ratio* ใช้สำหรับปรับค่าแอสเป็กเรโชของจอภาพให้มีสัดส่วนที่ถูกต้อง ฟังก์ชัน AspectRatio เป็นดังนี้

```
FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;
```

โปรแกรม 5.1 ถึง 5.4 ต่อไปนี้ จะแสดงการเลื่อนภาพ ที่มีลักษณะต่างๆ กัน โดยทุกโปรแกรมจะทำงานคล้ายกัน คือ แสดงภาพเริ่มต้นที่จอภาพ เมื่อกดคีย์ Enter ภาพนั้นจะเลื่อนไปอยู่ตำแหน่งใหม่ทันที ความแตกต่างที่เห็นได้ชัดของโปรแกรมเหล่านี้คือ การเรียกโพรซีเยอร์ SetPointer เพื่อสร้างภาพต่างๆ

ตัวอย่าง โปรแกรม 5.1 แสดงการเลื่อนภาพจุด จากตำแหน่ง 100,100 ไปยังตำแหน่งที่ห่างจาก X เท่ากับ 100 และห่างจาก Y เท่ากับ 100

โปรแกรม 5.1

```
PROGRAM TransPix;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
             Visible : Boolean;
             X       : Real;
             Y       : Real;
             Next    : Pointer;
          END;
```

```
VAR
  GraphDriver,GraphMode : Integer;
  Points       : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
  GetAspectRatio(Xasp,Yasp);
  AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics  : Pointer;
   Xc,Yc      : Integer;
BEGIN
  Graphics := Points;
  WHILE Graphics <> Nil DO
  BEGIN
    Xc  := Round(Graphics^.X);
    Yc  := Round(AspectRatio * Graphics^.Y);
    Yc  := GetMaxY - Yc;    {TrueY for turbo pascal}
    IF Graphics^.Visible THEN LineTo(Xc,Yc)
                         ELSE MoveTo(Xc,Yc);
    Graphics := Graphics^.Next;
  END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
```

```
   Color := GetColor;
   SetColor(GetBkColor);
   Draw(Points);
   SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
 VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Visible := VisibleC;
         X       := Xc;
         Y       := Yc
      END;
      Last := Ptr;
   END;

BEGIN
   Points := Nil;
   SetPointer(False, 100, 100);
   SetPointer(True,  100, 100);
   Last^.Next := Nil;
END;
```

```
PROCEDURE Translate(Points : Pointer; TransX,TransY : Real);
BEGIN
  WHILE Points <> Nil DO
  WITH Points^ DO
  BEGIN
    X := X + TransX;
    Y := Y + TransY;
    Points := Next
  END;
END;

BEGIN {MAIN}
  SetPointer(Points);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  Draw(Points); ReadLn;
  Blank(Points);
  Translate(Points,100,100);
  Draw(Points);
  ReadLn;
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 5.2 แสดงการเลื่อนภาพเส้นตรงขนาดความยาว 91 พิกเซล ที่มีจุดเริ่มต้นอยู่ที่ตำแหน่ง 10,10 ไปยังตำแหน่งที่ห่างจากเดิมเท่ากับ 90 พิก-เซลในทิศทางขนานกับแกน X

โปรแกรม 5.2

```
PROGRAM TransLine;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
              Visible : Boolean;
              X       : Real;
              Y       : Real;
              Next    : Pointer;
           END;
VAR
   GraphDriver,GraphMode : Integer;
   Points        : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
```

```
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                            ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
  Color := GetColor;
  SetColor(GetBkColor);
  Draw(Points);
  SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

  PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
  BEGIN
    New(Ptr);
    IF PointS <> Nil THEN Last^.Next := Ptr;
    IF PointS =  Nil THEN Points := Ptr;
    WITH Ptr^ DO
    BEGIN
       Visible := VisibleC;
       X       := Xc;
```

```
      Y        := Yc
    END;
    Last := Ptr;
  END;

BEGIN
  Points := Nil;
  SetPointer(False,  10, 10);
  SetPointer(True,  100,100);
  Last^.Next := Nil;
END;

PROCEDURE Translate(Points : Pointer; TransX,TransY : Real);
BEGIN
  WHILE Points <> Nil DO
  WITH Points^ DO
  BEGIN
    X := X + TransX;
    Y := Y + TransY;
    Points := Next
  END;
END;

BEGIN {MAIN}
  SetPointer(Points);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  Draw(Points); ReadLn;
  Blank(Points);
```

```
  Translate(Points,100,10);
  Draw(Points);    Readln;
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 5.3 แสดงการเลื่อนภาพสี่เหลี่ยมผืนผ้าขนาด 141 x 91 ซึ่งมีมุมล่างซ้ายอยู่ที่ตำแหน่ง 10,10 ไปยังตำแหน่งที่มุมล่างซ้ายเป็น 100,100

โปรแกรม 5.3

```
PROGRAM TransBox;
USES Graph;
TYPE
  Pointer = ^Pixel;
  Pixel = RECORD
            Visible : Boolean;
            X       : Real;
            Y       : Real;
            Next    : Pointer;
          END;
VAR
  GraphDriver,GraphMode : Integer;
  Points        : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
  GetAspectRatio(Xasp,Yasp);
```

```
   AspectRatio := Xasp/Yasp
END;


PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
   Color := GetColor;
   SetColor(GetBkColor);
   Draw(Points);
   SetColor(Color);
END;


PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;
```

```
PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
BEGIN
  New(Ptr);
  IF PointS <> Nil THEN Last^.Next := Ptr;
  IF PointS =  Nil THEN Points := Ptr;
  WITH Ptr^ DO
  BEGIN
    Visible := VisibleC;
    X       := Xc;
    Y       := Yc
  END;
  Last := Ptr;
END;

BEGIN
  Points := Nil;
  SetPointer(False,  10,  10);
  SetPointer(True,  150,  10);
  SetPointer(True,  150, 100);
  SetPointer(True,   10, 100);
  SetPointer(True,   10,  10);
  Last^.Next := Nil;
END;

PROCEDURE Translate(Points : Pointer; TransX,TransY : Real);
BEGIN
  WHILE Points <> Nil DO
  WITH Points^ DO
  BEGIN
    X := X + TransX;
```

```
      Y := Y + TransY;
      Points := Next
    END;
END;

BEGIN  {MAIN}
    SetPointer(Points);
    GraphDriver := Detect;
    InitGraph(GraphDriver,GraphMode,'');
    Draw(Points); ReadLn;
    Blank(Points);
    Translate(Points,100,100);
    Draw(Points);   Readln;
    CloseGraph
END.
```

ตัวอย่าง โปรแกรม 5.4 แสดงการเลื่อนภาพหัวลูกศรซึ่งอยู่บริเวณล่างซ้ายของจอภาพไปยังบริเวณขวาบนของจอภาพ

โปรแกรม 5.4

```
PROGRAM TransPol;
USES Graph;
TYPE
    Pointer = ^Pixel;
    Pixel = RECORD
                Visible : Boolean;
                X       : Real;
```

```
            Y          : Real;
            Next       : Pointer;
        END;

VAR
   GraphDriver,GraphMode : Integer;
   Points       : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc  := Round(Graphics^.X);
      Yc  := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc  := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;
```

```
PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
   Color := GetColor;
   SetColor(GetBkColor);
   Draw(Points);
   SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Visible := VisibleC;
         X       := Xc;
         Y       := Yc
      END;
      Last := Ptr;
   END;

BEGIN
   Points := Nil;
   SetPointer(False,  10,  10);
   SetPointer(True,  150,  80);
```

```
   SetPointer(True,   100, -80);
   SetPointer(True,    90,  10);
   SetPointer(True,    10,  10);
   SetPointer(False,   90,  10);
   SetPointer(True,    10, -80);
   Last^.Next := Nil;
END;

PROCEDURE Translate(Points : Pointer; TransX,TransY : Real);
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      X := X + TransX;
      Y := Y + TransY;
      Points := Next
   END;
END;

BEGIN  {MAIN}
   SetPointer(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Draw(Points);    ReadLn;   Blank(Points);
   Translate(Points,100,100);
   Draw(Points);    ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 5.5 แสดงการเลื่อนภาพหัวลูกศรซึ่งอยู่บริเวณล่างซ้ายของจอภาพไปยังบริเวณขวาบนของจอภาพ การเลื่อนมีลักษณะต่อเนื่องกัน จากจุดเริ่มต้นถึงจุดสุดท้าย โปรแกรมจะเริ่มด้วยการแสดงภาพลูกศร เมื่อกดคีย์ Enter ลูกศรจะเลื่อนไปทางขวาบน

โปรแกรม 5.5

```
PROGRAM TransMove;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
              Visible : Boolean;
              X       : Real;
              Y       : Real;
              Next    : Pointer;
           END;
VAR
   GraphDriver,GraphMode,i : Integer;
   Points       : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
```

```
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc  := Round(Graphics^.X);
      Yc  := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc  := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
   Color := GetColor;
   SetColor(GetBkColor);
   Draw(Points);
   SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
      New(Ptr);
```

```
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Visible := VisibleC;
         X       := Xc;
         Y       := Yc
      END;
         Last := Ptr;
   END;

BEGIN
   Points := Nil;
   SetPointer(False,  10,  10);
   SetPointer(True,  150,  80);
   SetPointer(True,  100, -80);
   SetPointer(True,   90,  10);
   SetPointer(True,   10,  10);
   SetPointer(False,  90,  10);
   SetPointer(True,   10, -80);
   Last^.Next := Nil;
END;

PROCEDURE Translate(Points : Pointer; TransX,TransY : Real);
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      X := X + TransX;
      Y := Y + TransY;
```

```
      Points := Next
    END;
END;

BEGIN  {MAIN}
   SetPointer(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Draw(Points); ReadLn;
   FOR I := 10 TO 20 DO
   BEGIN
      Blank(Points);
      Translate(Points,I,I);
      Draw(Points);
   END;
   Readln;
   CloseGraph
END.
```

5.1.2 การกำหนดสเกล หมายถึงการเปลี่ยนขนาดของภาพ ให้แตกต่างจากขนาดเดิม สมมติว่าขนาดของภาพตามแนวแกน X มีค่าเท่ากับ X และขนาดของภาพตามแนวแกน Y มีค่าเท่ากับ Y ถ้ากำหนดสเกลของภาพเป็นค่า Scale ขนาดของ X และ Y ใหม่ จะหาได้จากสมการต่อไปนี้

X = X * Scale

Y = Y * Scale

ถ้า Scale เป็นเรียล จะสามารถขยายภาพได้เมื่อกำหนดค่า Scale ให้มากกว่า 1 และสามารถย่อภาพได้เมื่อกำหนดค่า Scale อยู่ระหว่าง 0 ถึง 1 แต่ถ้า Scale เป็นอินติเจอร์ จะทำได้เฉพาะการขยายภาพเท่านั้น

กรณีที่ต้องการกำหนดสเกลในแนว X และ Y ให้มีค่าแตกต่างกัน เช่นกำหนดเป็น ScaleX และ ScaleY ขนาดของ X และ Y ใหม่จะหาได้จากสมการต่อไปนี้

X = X * ScaleX

Y = Y * ScaleY

ตัวอย่างโพรซีเยอร์ที่ใช้กำหนดสเกล โดยกำหนด ScaleX ให้แก่ X และกำหนด ScaleY ให้แก่ Y แสดงดังต่อไปนี้

```
PROCEDURE Scale(Points : Pointer; ScaleX,ScaleY : Real);
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      X  := X * ScaleX;
      Y  := Y * ScaleY;
      Points := Next
   END;
END;
```

โปรแกรม 5.6 ถึง 5.8 ต่อไปนี้ จะแสดงการกำหนดสเกลให้แก่ภาพต่างๆ โดยโปรแกรมจะทำงานคล้ายกันคือ แสดงภาพเริ่มต้นที่จอภาพ เมื่อกดคีย์ Enter ภาพจะขยายโตขึ้น และเมื่อกดคีย์ Enter อีกครั้งหนึ่ง ภาพจะมีขนาดเล็กลง

ตัวอย่าง โปรแกรม 5.6 แสดงภาพที่เกิดจากการกำหนดสเกลให้แก่เส้นตรงซึ่งมีขนาดเริ่มต้นเท่ากับ 91 แล้วกำหนดสเกล (X,Y) ให้เป็น 2,2 และ 0.225, 0.225 ตามลำดับ

โปรแกรม 5.6

```
PROGRAM ScaleLine;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
              Visible : Boolean;
              X       : Real;
              Y       : Real;
              Next    : Pointer;
           END;
VAR
   GraphDriver,GraphMode : Integer;
   Points      : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
```

```
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);

      Graphics := Graphics^.Next;
   END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
   Color := GetColor;
   SetColor(GetBkColor);
   Draw(Points);
   SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Visible := VisibleC;
```

```
      X         := Xc;
      Y         := Yc
    END;
    Last := Ptr;
  END;

BEGIN
  Points := Nil;
  SetPointer(False,  10, 10);
  SetPointer(True,  100,100);
  Last^.Next := Nil;
END;

PROCEDURE Scale(Points : Pointer; ScaleX,ScaleY : Real);
BEGIN
  WHILE Points <> Nil DO
  WITH Points^ DO
  BEGIN
    X  := X * ScaleX;
    Y  := Y * ScaleY;
    Points := Next
  END;
END;

BEGIN  {MAIN}
  SetPointer(Points);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  Draw(Points);    ReadLn;    Blank(Points);
  Scale(Points,2,2);
```

```
   Draw(Points);    Readln;    Blank(Points);
   Scale(Points,0.225,0.225);
   Draw(Points);    Readln;    Blank(Points);
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 5.7 แสดงภาพที่เกิดจากการกำหนดสเกลให้แก่ภาพสี่เหลี่ยมผืนผ้า ซึ่งเริ่มต้นด้วยขนาด 141 x 91 แล้วกำหนดสเกล (X,Y)ให้เป็น 2,2 และ 0.2,0.2 ตามลำดับ

โปรแกรม 5.7

```
PROGRAM ScaleBox;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
              Visible : Boolean;
              X       : Real;
              Y       : Real;
              Next    : Pointer;
           END;
VAR
   GraphDriver,GraphMode : Integer;
   Points       : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
```

```
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                         ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
```

```
      WITH Ptr^ DO
      BEGIN
         Visible  := VisibleC;
         X        := Xc;
         Y        := Yc
      END;
      Last := Ptr;
   END;

BEGIN
   Points := Nil;
   SetPointer(False,  10,  10);
   SetPointer(True,  150,  10);
   SetPointer(True,  150, 100);
   SetPointer(True,   10, 100);
   SetPointer(True,   10,  10);
   Last^.Next := Nil;
END;

PROCEDURE Scale(Points : Pointer; ScaleX,ScaleY : Real);
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      X  := X * ScaleX;
      Y  := Y * ScaleY;
      Points := Next
   END;
END;
```

```
BEGIN {MAIN}
  SetPointer(Points);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  Draw(Points); ReadLn;
  Scale(Points,2,2);
  Draw(Points); ReadLn;
  Scale(Points,0.2,0.2);
  Draw(Points);Readln;
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 5.8 แสดงภาพที่เกิดจากการกำหนดสเกลให้แก่ภาพหัวลูกศร เพื่อให้มีขนาดขยาย และย่อจากภาพเดิม ตามลำดับ

โปรแกรม 5.8

```
PROGRAM ScalePol;
USES Graph;
TYPE
  Pointer = ^Pixel;
  Pixel = RECORD
            Visible : Boolean;
            X       : Real;
            Y       : Real;
            Next    : Pointer;
          END;
VAR
  GraphDriver,GraphMode : Integer;
```

```
   Points        : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc       : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
   Color := GetColor;
   SetColor(GetBkColor);
```

```
  Draw(Points);
  SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

  PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
  BEGIN
    New(Ptr);
    IF PointS <> Nil THEN Last^.Next := Ptr;
    IF PointS =  Nil THEN Points := Ptr;
    WITH Ptr^ DO
    BEGIN
      Visible := VisibleC;
      X       := Xc;
      Y       := Yc
    END;
    Last := Ptr;
  END;

BEGIN
  Points := Nil;
  SetPointer(False,  10,  10);
  SetPointer(True,  150,  80);
  SetPointer(True,  100, -80);
  SetPointer(True,   90,  10);
  SetPointer(True,   10,  10);
  SetPointer(False,  90,  10);
  SetPointer(True,   10, -80);
```

```
   Last^.Next := Nil;
END;

PROCEDURE Scale(Points : Pointer; ScaleX,ScaleY : Real);
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      X   := X * ScaleX;
      Y   := Y * ScaleY;
      Points := Next
   END;
END;

BEGIN  {MAIN}
   SetPointer(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Draw(Points);    ReadLn;   Blank(Points);
   Scale(Points,1.5,1.5);
   Draw(Points);    ReadLn;   Blank(Points);
   Scale(Points,0.225,0.225);
   Draw(Points);    ReadLn;
   CloseGraph
END.
```

5.1.3 การหมุน หมายถึงการเปลี่ยนมุมของภาพให้แตกต่างจากเดิม สมมติว่ามีภาพอยู่ที่ X,Y ถ้าต้องการหมุนภาพไป Theta องศาในทิศทางตามเข็มนาฬิกา ค่าของ X,Y ใหม่จะหาได้จากสมการต่อไปนี้

X = X * Cos(Theta) + Y * Sin(Theta)

Y = -X * Sin(Theta) + Y * Cos(Theta)

ตามปกติพารามิเตอร์ของฟังก์ชั่น Sin และ Cos จะมีหน่วยเป็นเรเดียน ดังนั้นจึงต้องเปลี่ยนค่ามุมจากหน่วยองศาให้เป็นหน่วยเรเดียนด้วยสมการต่อไปนี้

Angle = 2 * Pi / 360 * Theta

โดยมุม Theta องศาจะมีค่าเป็น Angle เรเดียน

ตัวอย่างโพรซีเยอร์ที่ใช้สำหรับหมุนภาพไป Theta องศา ในทิศทางตามเข็มนาฬิกาแสดงดังต่อไปนี้

```
PROCEDURE Rotate(Points : Pointer; Theta : Real);
VAR XC,Angle   : Real;
BEGIN
  Angle := Theta * (Pi/180); {convert degree to radian}
  WHILE Points <> Nil DO
  WITH Points^ DO
  BEGIN
    XC :=  X * Cos(Angle) + Y * Sin(Angle);
    Y  := -X * Sin(Angle) + Y * Cos(Angle);
    X  := XC;
    Points := Next
  END
END;
```

โปรแกรม 5.9 ถึง 5.11 จะแสดงการหมุนภาพในทิศทางตามเข็มนาฬิกา โดยโปรแกรมจะทำงานคล้ายกันคือ แสดงภาพเริ่มต้นที่จอภาพ เมื่อกดคีย์ Enter ภาพจะหมุนไป 45 องศา และเมื่อกดคีย์ Enter อีกครั้งหนึ่งภาพจะหมุนไปอีก 45 องศา

ตัวอย่าง โปรแกรม 5.9 แสดงภาพที่เกิดจากการหมุนเส้นตรงซึ่งมีขนาดเท่ากับ 301 พิกเซล ที่อยู่ในแนวตั้ง โดยหมุนในทิศทางตามเข็มนาฬิกาครั้งละ 45 องศา

โปรแกรม 5.9

```
PROGRAM RotateLine;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
              Visible : Boolean;
              X       : Real;
              Y       : Real;
              Next    : Pointer;
           END;
VAR
   GraphDriver,GraphMode : Integer;
   Points       : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;
```

```
PROCEDURE Draw(Points : Pointer);
VAR Graphics  : Pointer;
   Xc,Yc      : Integer;
BEGIN
  Graphics := Points;
  WHILE Graphics <> Nil DO
  BEGIN
   Xc  := Round(Graphics^.X);
   Yc  := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
   Yc  := Round(AspectRatio * Yc);
   IF Graphics^.Visible THEN LineTo(Xc,Yc)
                       ELSE MoveTo(Xc,Yc);
   Graphics := Graphics^.Next;
  END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color ; Word;
BEGIN
  Color := GetColor;
  SetColor(GetBkColor);
  Draw(Points);
  SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

  PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
  BEGIN
```

```
    New(Ptr);
    IF PointS <> Nil THEN Last^.Next := Ptr;
    IF PointS =  Nil THEN Points := Ptr;
    WITH Ptr^ DO
    BEGIN
       Visible := VisibleC;
       X       := Xc;
       Y       := Yc
    END;
    Last := Ptr;
  END;

BEGIN
   Points  := Nil;
   SetPointer(False, 0, 0);
   SetPointer(True,  0, 300);
   Last^.Next := Nil;
END;

PROCEDURE Rotate(Points : Pointer; Theta : Real);
VAR XC,Angle   : Real;
BEGIN
   Angle := Theta * (Pi/180); {convert degree to radian}
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      XC :=  X * Cos(Angle) + Y * Sin(Angle);
      Y  := -X * Sin(Angle) + Y * Cos(Angle);
      X  := XC;
```

```
      Points := Next
    END
  END;

  BEGIN {MAIN}
    SetPointer(Points);
    GraphDriver := Detect;
    InitGraph(GraphDriver,GraphMode,'');
    Draw(Points);  ReadLn;  Blank(Points);
    Rotate(Points,45);
    Draw(Points);  Readln;  Blank(Points);
    Rotate(Points,45);
    Draw(Points);  Readln;
    CloseGraph
  END.
```

ตัวอย่าง โปรแกรม 5.10 แสดงภาพที่เกิดจากการหมุนภาพสี่เหลี่ยมผืนผ้าขนาด 101 x 51 พิกเซลไปในทิศทางตามเข็มนาฬิกา ครั้งละ 45 องศา

โปรแกรม 5.10

```
PROGRAM RotateBox;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
              Visible : Boolean;
              X       : Real;
              Y       : Real;
```

```
        Next      : Pointer;
      END;
VAR
   GraphDriver,GraphMode  : Integer;
   Points          : Pointer;
   X1, Y1, X2, Y2   : Integer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;
```

```
PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

  PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
  BEGIN
    New(Ptr);
    IF PointS <> Nil THEN Last^.Next := Ptr;
    IF PointS =  Nil THEN Points := Ptr;
    WITH Ptr^ DO
    BEGIN
      Visible := VisibleC;
      X       := Xc;
      Y       := Yc
    END;
    Last := Ptr;
  END;

BEGIN
  X1 := 0; Y1 := 100; X2 := X1 + 100; Y2 := Y1 + 150;
  Points := Nil;
  SetPointer(False,  X1,  Y1);
  SetPointer(True,   X2,  Y1);
  SetPointer(True,   X2,  Y2);
  SetPointer(True,   X1,  Y2);
  SetPointer(True,   X1,  Y1);
  Last^:Next := Nil;
END;

PROCEDURE Rotate(Points : Pointer; Theta : Real);
VAR XC,Angle  : Real;
```

```
BEGIN
   Angle := Theta * (Pi/180); {convert degree to radian}
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      XC :=  X * Cos(Angle) + Y * Sin(Angle);
      Y  := -X * Sin(Angle) + Y * Cos(Angle);
      X  := XC;
      Points := Next
   END
END;

BEGIN  {MAIN}
   SetPointer(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Draw(Points); ReadLn;
   Rotate(Points,45); Draw(Points); ReadLn;
   Rotate(Points,45); Draw(Points); ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 5.11 แสดงภาพที่เกิดจากการหมุนภาพหัวลูกศรไปในทิศทางตามเข็มนาฬิกา ครั้งละ 45 องศา

โปรแกรม 5.11

```
PROGRAM RotatePol;
USES Graph;
```

```
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
              Visible : Boolean;
              X        : Real;
              Y        : Real;
              Next     : Pointer;
           END;
VAR
   GraphDriver,GraphMode : Integer;
   Points       : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
```

```
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
   Color := GetColor;
   SetColor(GetBkColor);
   Draw(Points);
   SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Visible := VisibleC;
         X       := Xc;
         Y       := Yc
      END;
```

```
      Last := Ptr;
   END;

BEGIN
   Points := Nil;
   SetPointer(False,  100,  300);
   SetPointer(True,  250,  380);
   SetPointer(True,  200,  200);
   SetPointer(True,   190,  300);
   SetPointer(True,   100,  300);
   SetPointer(False,  190,  300);
   SetPointer(True,   100,  220);
   Last^.Next := Nil;
END;

PROCEDURE Rotate(Points : Pointer; Theta : Real);
VAR XC,Angle  : Real;
BEGIN
   Angle := Theta * (Pi/180); {convert degree to radian}
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      XC :=  X * Cos(Angle) + Y * Sin(Angle);
      Y  := -X * Sin(Angle) + Y * Cos(Angle);
      X  := XC;
      Points := Next
   END
END;
```

```
BEGIN  {MAIN}
   SetPointer(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Draw(Points);           ReadLn;  Blank(Points);
   Rotate(Points,45);  Draw(Points);   ReadLn;
   Rotate(Points,45);  Draw(Points);   ReadLn;
   CloseGraph
END.
```

5.1.4 การเบ้ หมายถึงการเปลี่ยนลักษณะของภาพให้ผิดเพี้ยนไปจากภาพเดิม สมมติว่า มีภาพอยู่ที่ X,Y ถ้าต้องการเบ้ภาพในแนว X ด้วยค่า Shear โดย Shear เป็นข้อมูลชนิดเรียล จะหาค่า X ใหม่ได้จากสมการต่อไปนี้

X = X + Shear * Y

และถ้าต้องการเบ้ภาพในแนว Y ด้วยค่า Shear เช่นเดียวกัน จะหาค่า Y ใหม่ได้จากสมการต่อไปนี้

Y = Y + Shear * X

กรณีที่ต้องการกำหนดค่า Shear ในแนว X และ Y ให้มีค่าแตกต่างกัน เช่น กำหนดเป็น ShearX และ ShearY ค่าของ X และ Y ใหม่จะหาได้จากสมการต่อไปนี้

X = X + ShearX * Y

Y = Y + ShearY * X

ตัวอย่างโพรซีเยอร์ที่ใช้สำหรับการเบ้ภาพโดยกำหนด ShearX ให้แก่ X และกำหนด ShearY ให้แก่ Y แสดงดังต่อไปนี้

```
PROCEDURE Shear(Points : Pointer; ShearX,ShearY : Real);
VAR TempX : Real;
BEGIN
  WHILE Points <> Nil DO
  WITH Points^ DO
  BEGIN
    TempX := X;     { store x value in tempx }
    X := X + ShearX * Y;
    Y := Y + ShearY * TempX;
    Points := Next
  END;
END;
```

โปรแกรม 5.12 และ 5.13 จะแสดงการเบ้ภาพ โดยโปรแกรมจะทำงานคล้ายกันคือแสดงภาพเริ่มต้นที่จอภาพ เมื่อกดคีย์ Enter ครั้งแรก ภาพจะเบ้ไปในแนว X และเมื่อกดคีย์ Enter ครั้งที่สอง ภาพจะเบ้ไปในแนว Y และเมื่อกดคีย์ Enter ครั้งที่สาม ภาพจะเบ้ไปในแนว X และ Y พร้อมกัน

ตัวอย่าง โปรแกรม 5.12 แสดงภาพที่เกิดจากการเบ้ภาพสี่เหลี่ยมผืนผ้าขนาด 141 x 91 พิกเซล โดยการเบ้ในแนว X แนว Y และแนว X,Y ตามลำดับ

โปรแกรม 5.12

```
PROGRAM ShearBox;
USES Graph;
TYPE
```

```
  Pointer = ^Pixel;
  Pixel = RECORD
              Visible : Boolean;
              X         : Real;
              Y         : Real;
              Next      : Pointer;
          END;
VAR
   GraphDriver,GraphMode : Integer;
   Points        : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc       : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
```

```
      Graphics := Graphics^.Next;
    END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
   Color := GetColor;
   SetColor(GetBkColor);
   Draw(Points);
   SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Visible := VisibleC;
         X       := Xc;
         Y       := Yc
      END;
      Last := Ptr;
   END;
```

```
BEGIN
   Points := Nil;
   SetPointer(False,  10,  10);
   SetPointer(True,  150,  10);
   SetPointer(True,  150, 100);
   SetPointer(True,   10, 100);
   SetPointer(True,   10,  10);
   Last^.Next := Nil;
END;

PROCEDURE Shear(Points : Pointer; ShearX,ShearY : Real);
VAR TempX : Real;
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      TempX := X;    { store x value in tempx }
      X := X + ShearX * Y;
      Y := Y + ShearY * TempX;
      Points := Next
   END;
END;

BEGIN {MAIN}
   SetPointer(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Draw(Points); ReadLn;  Blank(Points);
   Shear(Points,0.5,0);    Draw(Points); ReadLn; Blank(Points);
   Shear(Points,0,0.5);    Draw(Points); ReadLn; Blank(Points);
```

```
  Shear(Points,0.5,0.5); Draw(Points); ReadLn; Blank(Points);
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 5.13 แสดงภาพที่เกิดจากการเบ้ภาพหัวลูกศร โดยการเบ้ในแนว X แนว Y และแนว X,Y ตามลำดับ

โปรแกรม 5.13

```
PROGRAM ShearPol;
USES Graph;
TYPE
  Pointer = ^Pixel;
  Pixel = RECORD
            Visible : Boolean;
            X       : Real;
            Y       : Real;
            Next    : Pointer;
          END;
VAR
  GraphDriver,GraphMode : Integer;
  Points        : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
  GetAspectRatio(Xasp,Yasp);
  AspectRatio := Xasp/Yasp
END;
```

```
PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;

PROCEDURE Blank(Points : Pointer);
VAR Color : Word;
BEGIN
   Color := GetColor;
   SetColor(GetBkColor);
   Draw(Points);
   SetColor(Color);
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
```

```
    New(Ptr);
    IF PointS <> Nil THEN Last^.Next := Ptr;
    IF PointS =  Nil THEN Points := Ptr;
    WITH Ptr^ DO
    BEGIN
      Visible := VisibleC;
      X       := Xc;
      Y       := Yc
    END;
    Last := Ptr;
  END;

BEGIN
  Points := Nil;
  SetPointer(False,  10,  10);
  SetPointer(True,  150,  80);
  SetPointer(True,  100, -80);
  SetPointer(True,   90,  10);
  SetPointer(True,   10,  10);
  SetPointer(False,  90,  10);
  SetPointer(True,   10, -80);
  Last^.Next := Nil;
END;

PROCEDURE Shear(Points : Pointer; ShearX,ShearY : Real);
VAR TempX : Real;
BEGIN
  WHILE Points <> Nil DO
  WITH Points^ DO
  BEGIN
```

```
      TempX := X;     { store x value in tempx }
      X := X + ShearX * Y;
      Y := Y + ShearY * TempX;
      Points := Next
   END;
END;

BEGIN  {MAIN}
   SetPointer(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   Draw(Points);    ReadLn;  Blank(Points);
   Shear(Points,0.5,0);     Draw(Points); ReadLn; Blank(Points);
   Shear(Points,0,0.5);     Draw(Points); ReadLn; Blank(Points);
   Shear(Points,0.5,0.5); Draw(Points); ReadLn; Blank(Points);
   CloseGraph
END.
```

5.2 การแปลงภาพหลายขั้น

การแปลงภาพหลายขั้น หมายถึงการแปลงภาพที่มีหลายขั้นตอน ซึ่งเป็นการนำวิธีการแปลงภาพจากการแปลงภาพขั้นเดียว มาปรับปรุงใช้ให้ตรงตามความต้องการ โดยอาจจะต้องเพิ่มเติมวิธีดำเนินการกับภาพบางอย่างเข้ามาด้วย เนื่องจากการแปลงภาพประเภทนี้เป็นเรื่องของการคิดดัดแปลงเพื่อนำไปใช้งาน จึงประกอบด้วยวิธีการที่ไม่จำกัดจำนวน ในที่นี้จะนำวิธีการแปลงภาพหลายขั้นมาเป็นตัวอย่างเพียง 7 ตัวอย่าง ดังต่อไปนี้

5.2.1 การแสดงภาพซ้ำในแนวนอน เป็นการนำภาพที่มีอยู่แล้วมาแสดงใหม่ โดยจัดให้แสดงในลักษณะเรียงต่อกันไปในแนวนอนมีทิศทางจากซ้ายไปขวา โพรซีเยอร์

RepeatX ต่อไปนี้จะทำหน้าที่ดังกล่าว ค่าที่จะต้องกำหนดให้แก่โพรซีเยอร์คือ ระยะห่างระหว่างแต่ละภาพ (Spacing) และจำนวนภาพที่ต้องการแสดง (PicNumber)

```
PROCEDURE RepeatX(Spacing : Real; PicNumber : Integer);
VAR NextX     : Real;
    Count : Integer;
BEGIN
   NextX := 0;
   FOR Count := 1 TO PicNumber DO
   BEGIN
      Translate(PointS, NextX, 0);
      Draw(Points);
      NextX := Spacing;
   END;
END;
```

ตัวอย่าง โปรแกรม 5.14 แสดงภาพซ้ำในแนวนอนภาพที่แสดงเป็นภาพสามเหลี่ยมซึ่งมีลักษณะคล้ายใบเรือของเรือใบขนาดเล็ก จำนวนภาพมีรวม 10 ภาพ

โปรแกรม 5.14

```
PROGRAM RepeatPictureX;
USES Graph;
TYPE
  Pointer = ^Pixel;
  Pixel = RECORD
```

```
            Visible  : Boolean;
            X          : Real;
            Y          : Real;
            Next       : Pointer;
         END;

VAR
   GraphDriver,GraphMode  : Integer;
   Points          : Pointer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc          : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                                    ELSE MoveTo(Xc,Yc);
```

```
    Graphics := Graphics^.Next;
  END;
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

  PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
  BEGIN
    New(Ptr);
    IF PointS <> Nil THEN Last^.Next := Ptr;
    IF PointS = Nil THEN Points := Ptr;
    WITH Ptr^ DO
    BEGIN
      Visible := VisibleC;
      X       := Xc;
      Y       := Yc
    END;
    Last := Ptr;
  END;

BEGIN
  Points := Nil;
  SetPointer(False,120,190);
  SetPointer(True,  95,270);
  SetPointer(True, 120,300);
  SetPointer(True, 120,190);
  Last^.Next := Nil;
END;
```

```
PROCEDURE Translate(Points : Pointer; TransX,TransY : Real);
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      X := X + TransX;
      Y := Y + TransY;
      Points := Next
   END;
END;

PROCEDURE RepeatX(Spacing : Real; PicNumber : Integer);
VAR NextX     : Real;
    Count : Integer;
BEGIN
   NextX := 0;
   FOR Count := 1 TO PicNumber DO
   BEGIN
      Translate(PointS, NextX, 0);
      Draw(Points);
      NextX := Spacing;
   END;
END;

BEGIN  {MAIN}
   SetPointer(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   RepeatX(50,10);
```

```
  ReadLn;
  CloseGraph
END.
```

5.2.2 การหมุนภาพรอบจุดใด ๆ การหมุนภาพที่ได้กล่าวมาแล้ว เป็นการหมุนภาพโดยมีจุดศูนย์กลางอยู่ที่จุดกำเนิด (origin) ของระบบโคออร์ดิเนตนั้น กรณีที่ต้องการหมุนภาพโดยใช้จุดอื่นเป็นจุดศูนย์กลางก็สามารถทำได้ด้วยหลักการต่อไปนี้

สมมติจุดศูนย์กลางการหมุนภาพเป็น XC,YC ต้องการหมุนภาพไป Theta องศา วิธีการหมุนภาพจะมี 3 ขั้นดังนี้

1. เลื่อนจุดศูนย์กลางใหม่ (XC,YC) ไปที่จุดกำเนิดของระบบโคออร์ดิเนตนั้น โดยใช้สเตตเมนต์

   Translate (Points, -CX, -CY)

2. หมุนภาพไป Theta องศา โดยใช้สเตตเมนต์

   Rotate (Points, Theta)

3. เลื่อนภาพกลับไปตำแหน่งเดิมด้วยสเตตเมนต์

   TransLate (Points, CX, CY)

ไพรซีเยอร์ RotateXY สำหรับการหมุนภาพรอบจุด XC,YC เป็นดังนี้

```
PROCEDURE RotateXY(Points : Pointer; CX,CY,Theta : Real);
BEGIN
  Translate(Points,-CX,-CY);
  Rotate(Points,Theta);
  Translate(Points,CX,CY);
END;
```

5.2.3 การสร้างภาพรอบจุดใด ๆ ภาพบางประเภทเกิดจากส่วนย่อยๆจำนวนหนึ่งของภาพ โดยการจัดเรียงส่วนย่อยๆ เหล่านั้นให้อยู่รอบจุดจุดหนึ่ง เช่น ภาพใบพัดของพัดลม ใบพัดของกังหันลม หลักการสร้างภาพประเภทนี้มี 2 ขั้นตอน ดังต่อไปนี้

1. สร้างส่วนย่อยของภาพจำนวน 1 ภาพ โดยการกำหนดค่า X,Y ต่างๆ ในโพรซีเยอร์ SetPointer

2. สร้างภาพย่อยรอบจุดที่กำหนด โดยให้มีจำนวนภาพตามที่ต้องการ ด้วยโพรซีเยอร์ RotateXY

ตัวอย่าง โปรแกรม 5.15 แสดงภาพกังหันลม ซึ่งเป็นภาพที่เกิดจากภาพสามเหลี่ยมขนาดเล็ก 1 ภาพ นำมาแสดงซ้ำที่ตำแหน่งต่างๆรอบจุดจุดหนึ่ง จำนวน 16 ครั้ง ทำให้มองเห็นเป็นภาพกังหันลมมีลักษณะกลม

โปรแกรม 5.15

```
PROGRAM RepeatPictureRotate;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
              Visible : Boolean;
              X       : Real;
              Y       : Real;
              Next    : Pointer;
           END;
VAR
   GraphDriver,GraphMode : Integer;
   Points       : Pointer;
```

```
FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc  := Round(Graphics^.X);
      Yc  := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc  := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
```

```
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Visible := VisibleC;
         X        := Xc;
         Y        := Yc
      END;
      Last := Ptr;
   END;

BEGIN
   Points := Nil;
   SetPointer(False,120,190);
   SetPointer(True,  95,270);
   SetPointer(True, 120,300);
   SetPointer(True, 120,190);
   Last^.Next := Nil;
END;

PROCEDURE Translate(Points : Pointer; TransX,TransY : Real);
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      X := X + TransX;
      Y := Y + TransY;
      Points := Next
   END;
END;
```

```
PROCEDURE Rotate(Points : Pointer; Theta : Real);
VAR XC,Angle   : Real;
BEGIN
  Angle := Theta * (Pi/180); {convert degree to radian}
  WHILE Points <> Nil DO
  WITH Points^ DO
  BEGIN
     XC :=  X * Cos(Angle) + Y * Sin(Angle);
     Y  := -X * Sin(Angle) + Y * Cos(Angle);
     X  := XC;
     Points := Next
  END
END;

PROCEDURE RotateXY(Points : Pointer; CX,CY,Theta : Real);
BEGIN
  Translate(Points,-CX,-CY);
  Rotate(Points,Theta);
  Translate(Points,CX,CY);
END;

PROCEDURE RepeatRotate(PicNumber : Integer);
VAR Increment, Angle : Real;
   Count : Integer;
BEGIN
  Increment := 360 / PicNumber;
  Angle := 0;
  FOR Count := 1 TO PicNumber DO
  BEGIN
     RotateXY(Points,120,190,Angle);
```

```
    Draw(Points);
    Angle := Angle + Increment;
  END;
END;

BEGIN  {MAIN}
  SetPointer(Points);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  RepeatRotate(16); ReadLn;
  CloseGraph
END.
```

5.2.4 การสร้างภาพสะท้อนแบบบน-ล่าง (Reflecty) ตัวอย่างลักษณะภาพสะท้อนแบบนี้ได้แก่การดูต้นไม้ริมฝั่งน้ำ ซึ่งเราจะเห็นทั้งภาพจริงของต้นไม้ และภาพที่สะท้อนขึ้นมาจากน้ำ ไพรซีเยอร์ ReflectY สำหรับสร้างภาพสะท้อนแบบบน-ล่างเป็นดังนี้

```
PROCEDURE ReflectY(Points : Pointer; TransY : Real);
BEGIN
  Translate(Points,0,-TransY);
  Scale(Points,1,-1);
  Translate(Points,0,TransY);
END;
```

พารามิเตอร์ TransY ใช้กำหนดตำแหน่งของภาพสะท้อนตามแนวแกน Y ถ้าค่าน้อย ภาพจะอยู่ด้านล่างของจอภาพ แต่ถ้ามีค่ามากภาพจะอยู่ด้านบนของจอภาพ

การกำหนดสเกลในสเตตเมนต์ Scale(Points, 1, -1) มีเงื่อนไขดังนี้

ค่า 1 หมายถึงภาพสะท้อนมีความกว้างเท่าภาพเดิม ถ้าค่ามากกว่า 1 ภาพสะท้อนจะกว้างกว่าภาพเดิม และถ้าค่าน้อยกว่า 1 ภาพสะท้อนจะแคบกว่าภาพเดิม

ค่า -1 หมายถึงภาพสะท้อนมีความสูงเท่าภาพเดิม ถ้าค่ามากกว่า -1 ภาพสะท้อนจะต่ำกว่าภาพเดิม และถ้าค่าน้อยกว่า -1 ภาพสะท้อนจะสูงกว่าภาพเดิม

5.2.5 การสร้างภาพสะท้อนแบบซ้าย-ขวา (Reflectx) ตัวอย่างลักษณะภาพสะท้อนแบบนี้ได้แก่การดูภาพจากกระจกเงา ซึ่งเราจะเห็นภาพกลับซ้าย-ขวา เมื่อเปรียบเทียบกับของจริง โพรซีเยอร์ ReflectX สำหรับสร้างภาพสะท้อนแบบซ้าย-ขวาเป็นดังนี้

```
PROCEDURE ReflectX(Points : Pointer; TransX : Real);
BEGIN
  Translate(Points,-TransX,0);
  Scale(Points,-1,1);
  Translate(Points,TransX,0);
END;
```

พารามิเตอร์ TransX ใช้กำหนดตำแหน่งของภาพสะท้อนตามแนวแกน X ถ้าค่าน้อย ภาพจะอยู่ทางด้านซ้ายของจอภาพ แต่ถ้ามีค่ามากภาพจะอยู่ทางด้านขวาของจอภาพ

การกำหนดสเกลในสเตตเมนต์ Scale(Points, -1, 1) มีเงื่อนไขดังนี้

ค่า -1 หมายถึงภาพสะท้อนมีความกว้างเท่าภาพเดิม ถ้าค่ามากกว่า -1 ภาพสะท้อนจะแคบกว่าภาพเดิม และถ้าค่าน้อยกว่า -1 ภาพสะท้อนจะกว้างกว่าภาพเดิม

ค่า 1 หมายถึงภาพสะท้อนมีความสูงเท่าภาพเดิม ถ้าค่ามากกว่า 1 ภาพสะท้อนจะสูงกว่าภาพเดิม และถ้าค่าน้อยกว่า 1 ภาพสะท้อนจะต่ำกว่าภาพเดิม

5.2.6 การสร้างภาพสะท้อนซ้ำรอบจุดใด ๆ เป็นการแสดงภาพ 2 วิธีการต่อเนื่องกันคือ สร้างภาพสะท้อนแบบซ้าย-ขวาจากภาพหนึ่ง แล้วแสดงภาพนั้นให้เรียงตามลำดับรอบจุดจุดหนึ่ง โพรซีเยอร์ที่ใช้มีดังนี้

1. ReflectX ใช้สำหรับสะท้อนภาพในแนวซ้าย-ขวา

2. Bilateral ใช้สำหรับแสดงภาพเดิมและภาพสะท้อน โดยมีการจัดระยะห่างระหว่างภาพคู่นี้ให้ได้ตามที่กำหนด

3. RepeatReflect ใช้สำหรับควบคุมการแสดงภาพให้อยู่รอบจุดที่กำหนด

ตัวอย่าง โปรแกรม 5.16 ใช้แสดงภาพสะท้อนซ้ำรอบจุดใดๆ ภาพต้นแบบเป็นภาพสามเหลี่ยมลักษณะคล้ายใบเรือ ทำภาพนี้ให้สะท้อนในแนวซ้าย-ขวา และอยู่ติดกับภาพต้นแบบ ซึ่งจะเกิดเป็นภาพคล้ายกลีบดอกไม้ เมื่อนำภาพกลีบดอกไม้มาแสดงซ้ำรอบจุดจุดหนึ่ง จะเกิดภาพที่มองดูคล้ายดอกไม้ 1 ดอก

โปรแกรม 5.16

```
PROGRAM RepeatPictureReflect;
USES Graph;
TYPE
  Pointer = ^Pixel;
  Pixel = RECORD
            Visible : Boolean;
            X       : Real;
            Y       : Real;
```

```
            Next      : Pointer;
         END;
VAR
   GraphDriver,GraphMode : Integer;
   Points         : Pointer;
   FirstX,FirstY  : Integer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
   BEGIN
      Xc   := Round(Graphics^.X);
      Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
      Yc   := Round(AspectRatio * Yc);
      IF Graphics^.Visible THEN LineTo(Xc,Yc)
                           ELSE MoveTo(Xc,Yc);
      Graphics := Graphics^.Next;
   END;
END;
```

```
PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

   PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
   BEGIN
      New(Ptr);
      IF PointS <> Nil THEN Last^.Next := Ptr;
      IF PointS =  Nil THEN Points := Ptr;
      WITH Ptr^ DO
      BEGIN
         Visible := VisibleC;
         X       := Xc;
         Y       := Yc
      END;
      Last := Ptr;
   END;

BEGIN
   Points := Nil;
   SetPointer(False,FirstX,      FirstY);
   SetPointer(True, FirstX-25, FirstY+80);
   SetPointer(True, FirstX,      FirstY+110);
   SetPointer(True, FirstX,      FirstY);
   Last^.Next := Nil;
END;

PROCEDURE Translate(Points : Pointer; TransX,TransY : Real);
BEGIN
   WHILE Points <> Nil DO
```

```
    WITH Points^ DO
    BEGIN
      X := X + TransX;
      Y := Y + TransY;
      Points := Next
    END;
END;

PROCEDURE Scale(Points : Pointer; ScaleX,ScaleY : Real);
BEGIN
    WHILE Points <> Nil DO
    WITH Points^ DO
    BEGIN
      X  := X * ScaleX;
      Y  := Y * ScaleY;
      Points := Next
    END;
END;

PROCEDURE Rotate(Points : Pointer; Theta : Real);
VAR XC,Angle  : Real;
BEGIN
    Angle := Theta * (Pi/180); {convert degree to radian}
    WHILE Points <> Nil DO
    WITH Points^ DO
    BEGIN
      XC :=  X * Cos(Angle) + Y * Sin(Angle);
      Y  := -X * Sin(Angle) + Y * Cos(Angle);
      X  := XC;
```

```
      Points := Next
    END
END;

PROCEDURE RotateXY(Points : Pointer; CX,CY,Theta : Real);
BEGIN
    Translate(Points,-CX,-CY);
    Rotate(Points,Theta);
    Translate(Points,CX,CY);
END;

PROCEDURE ReflectX(Points : Pointer; TransX : Real);
BEGIN
    Translate(Points,-TransX,0);
    Scale(Points,-1,1);
    Translate(Points,TransX,0);
END;

PROCEDURE Bilateral(Separate : Real);
BEGIN
    Draw(Points);
    ReflectX(Points,Separate);
    Draw(Points);
END;

PROCEDURE RepeatReflect(Separate : Real; PicNumber : Integer);
VAR Increment, Angle : Real;
    Count : Integer;
BEGIN
    Increment := 360 / PicNumber;
```

```
   Angle := 0;
   FOR Count := 1 TO PicNumber DO
   BEGIN
      RotateXY(Points,FirstX,FirstY,Angle);
      Bilateral(Separate);
      Angle := Angle + Increment;
   END;
END;

BEGIN  {MAIN}
   FirstX := 350;
   FirstY := 150;
   SetPointer(Points);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   RepeatReflect(FirstX,7); ReadLn;
   CloseGraph
END.
```

5.2.7 การแสดงภาพซ้ำแบบลดขนาด เป็นการแสดงภาพหนึ่งซ้ำๆ กันโดยมีการลดขนาดภาพลงเรื่อยๆ

ตัวอย่าง โปรแกรม 5.17 แสดงภาพสี่เหลี่ยมจัตุรัสซ้ำกันหลายภาพโดยภาพแรกมีขนาดใหญ่ที่สุด ภาพต่อๆ มาจะถูกลดขนาดลงเรื่อยๆ โพรซีเยอร์ที่ทำหน้าที่นี้ คือ RepeatReflectX ค่าที่กำหนดให้โพรซีเยอร์มี 2 ค่าคือ อัตราการลดขนาด (Decrement) ซึ่งเป็นค่าจุดทศนิยม และค่าจำนวนภาพที่ต้องการให้แสดง (PicNumber) เป็นจำนวนเต็ม

โปรแกรม 5.17

```
PROGRAM RepeatPictureDecrement;
USES Graph;
TYPE
   Pointer = ^Pixel;
   Pixel = RECORD
              Visible : Boolean;
              X       : Real;
              Y       : Real;
              Next    : Pointer;
           END;
VAR
   GraphDriver,GraphMode : Integer;
   Points         : Pointer;
   FirstX,FirstY : Integer;

FUNCTION AspectRatio : Real;
VAR  Xasp,Yasp : Word;
BEGIN
   GetAspectRatio(Xasp,Yasp);
   AspectRatio := Xasp/Yasp
END;

PROCEDURE Draw(Points : Pointer);
VAR Graphics   : Pointer;
    Xc,Yc      : Integer;
BEGIN
   Graphics := Points;
   WHILE Graphics <> Nil DO
```

```
  BEGIN
     Xc   := Round(Graphics^.X);
     Yc   := GetMaxY-(Round(Graphics^.Y)); {trueY for turbo pascal}
     Yc   := Round(AspectRatio * Yc);
     IF Graphics^.Visible THEN LineTo(Xc,Yc)
                          ELSE MoveTo(Xc,Yc);
     Graphics := Graphics^.Next;
  END;
END;

PROCEDURE SetPointer(VAR Points : Pointer);
VAR Last,Ptr : Pointer;

  PROCEDURE SetPointer(VisibleC : Boolean; XC,YC : Real);
  BEGIN
     New(Ptr);
     IF PointS <> Nil THEN Last^.Next := Ptr;
     IF PointS =  Nil THEN Points := Ptr;
     WITH Ptr^ DO
     BEGIN
        Visible := VisibleC;
        X       := Xc;
        Y       := Yc
     END;
     Last := Ptr;
  END;

BEGIN
  Points := Nil;
  SetPointer(False,FirstX,      FirstY);
```

```
   SetPointer(True, FirstX+200, FirstY);
   SetPointer(True, FirstX+200, FirstY+200);
   SetPointer(True, FirstX,     FirstY+200);
   SetPointer(True, FirstX,     FirstY);
   Last^.Next := Nil;
END;

PROCEDURE Scale(Points : Pointer; ScaleX,ScaleY : Real);
BEGIN
   WHILE Points <> Nil DO
   WITH Points^ DO
   BEGIN
      X  := X * ScaleX;
      Y  := Y * ScaleY;
      Points := Next
   END;
END;

PROCEDURE RepeatPicture(Decrement : Real; RepeatNumber : Integer);
VAR Length : Real;
    Count  : Integer;
BEGIN
   Length := 1;
   Draw(points);
   FOR Count := 1 TO RepeatNumber-1 DO
   BEGIN
      Length := Length - Decrement;
      Scale(Points,Length,Length);
      Draw(Points);
```

```
    Length := Length + Decrement;
  END;
END;

BEGIN  {MAIN}
  FirstX := 450;
  FirstY := 100;
  SetPointer(Points);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  RepeatPicture(0.1,20);
  ReadLn;
  CloseGraph
END.
```

6

การแสดงภาพจากไฟล์ข้อมูลภาพแบบบิตแมพ

มีโปรแกรมกราฟิกส์ซึ่งเป็นโปรแกรมสำเร็จรูปหลายโปรแกรม ที่ใช้วิธีการสร้างภาพแบบบิตแมพและบันทึกข้อมูลภาพแบบบิตแมพไว้ในไฟล์ เราสามารถอ่านข้อมูลจากไฟล์ดังกล่าวมาแสดงเป็นภาพบนจอภาพได้ ด้วยโปรแกรมที่เราเขียนขึ้นเอง ซึ่งมีหลักการและวิธีดำเนินการดังต่อไปนี้

6.1 การแสดงภาพจากไฟล์ที่ทราบขอบเขตของภาพ

ตามปกติข้อมูลภาพที่เก็บอยู่ในไฟล์มักจะเป็นข้อมูลของภาพที่อยู่ในกรอบสี่เหลี่ยมซึ่งมีขนาดใดขนาดหนึ่ง ดังนั้นความกว้างของภาพจึงหมายถึง จำนวนจุดหรือจำนวนพิกเซลทั้งหมดที่มีอยู่ในแนวนอน และความสูงของภาพจะหมายถึงจำนวนจุดทั้งหมดที่มีอยู่ในแนวตั้ง ซึ่งถ้าทราบค่าความกว้างและความสูงของภาพ เราก็จะสามารถเขียนโปรแกรมแสดงภาพนั้นได้

โปรแกรมสำเร็จรูปที่ระบุขอบเขตของภาพ ซึ่งเป็นที่รู้จักกันดีได้แก่ โปรแกรมพรินต์มาสเตอร์พลัส (PrintMaster Plus) และโปรแกรมเดอะนิวพรินต์ชอพ (The New Print Shop) โปรแกรมพรินต์มาสเตอร์พลัสเป็นของบริษัท Kyocera Unison, Inc. ออกจำหน่ายในปี ค.ศ. 1988 โปรแกรมเดอะนิวพรินต์ชอพเป็นของบริษัท Pixellite Software, Inc. ออกจำหน่ายในปี ค.ศ. 1989

โปรแกรมที่นำมาเป็นตัวอย่างในบทนี้ได้แก่ โปรแกรมพรินต์มาสเตอร์พลัส หน้าที่หลักของโปรแกรมนี้คือ การพิมพ์ภาพและอักษรในแบบกราฟิกส์ ซึ่งสามารถพิมพ์งานได้หลายชนิด เช่น ภาพโปสเตอร์ ตัวหนังสือขนาดใหญ่ การ์ดสำหรับโอกาสและเทศกาลต่างๆ นอกจากนั้น โปรแกรมยังมีกราฟิกเอดิเตอร์สำหรับใช้สร้างภาพใหม่ขึ้นเองและแก้ไขเพิ่มเติมภาพที่มีอยู่ให้ตรงกับความต้องการได้ด้วย

ภาพที่ใช้ในโปรแกรมพรินต์มาสเตอร์พลัส อาจจะนำมาจากไฟล์ที่เราสร้างขึ้นเอง หรือนำมาจากไฟล์ภาพซึ่งผู้เขียนโปรแกรมได้จัดเตรียมมาให้เรียบร้อยแล้วก็ได้ ในที่นี้จะกล่าวถึงเฉพาะไฟล์ภาพประเภทหลัง

ไฟล์เก็บภาพที่มีมาพร้อมกับโปรแกรมพรินต์มาสเตอร์พลัสคือ ไฟล์ชื่อ STANDARD.SHP ซึ่งเป็นไฟล์ขนาด 70394 ไบต์ ในไฟล์มีภาพ 122 ภาพ แต่ละภาพมีขนาดเท่ากันคือ กว้าง 88 พิกเซลและสูง 52 พิกเซล ข้อมูลของแต่ละภาพเท่ากับ 577 ไบต์ ตามปกติโปรแกรมพรินต์มาสเตอร์พลัสจะทำงานร่วมกับไฟล์ STANDARD.SHP โดยพรินต์มาสเตอร์พลัสจะทราบตำแหน่งของข้อมูลแต่ละภาพและอ่านข้อมูลนั้นมาแสดงที่จอภาพได้ทันที

การเขียนโปรแกรมเพื่อนำภาพจากไฟล์ STANDARD.SHP มาแสดงที่จอภาพสามารถทำได้ทั้งการทำงานร่วมกับไฟล์ STANDARD.SHP ทั้งไฟล์ หรือการเลือกเอาเฉพาะข้อมูลของภาพที่ต้องการมาเก็บไว้ในไฟล์ใหม่ไฟล์ละ 1 ภาพก็ได้ ในที่นี้จะเริ่มด้วยการเขียนโปรแกรมสำหรับแสดงภาพจากไฟล์ที่มี 1 ภาพก่อน เพราะมีความพร้อมและสามารถทำความเข้าใจได้ง่ายกว่า

6.1.1 การแสดงภาพจากไฟล์ที่มีภาพเดียว ไฟล์ข้อมูลภาพที่นำมาเป็นตัวอย่างในหัวข้อนี้อาจจะใช้ไฟล์ BIRD.GTP และ PUPPY.GTP ที่เราได้สร้างไว้ในบทที่ 3 หรือจะแยกเอาภาพใดภาพหนึ่งออกมาจากไฟล์ STANDARD.SHP ก็ได้

วิธีการแยกข้อมูลภาพ 1 ภาพจากไฟล์ STANDARD.SHP มาเก็บไว้ในไฟล์ใหม่ซึ่งเป็นไฟล์ขนาด 577 ไบต์ สามารถทำได้ 2 วิธีคือ โดยการรันโปรแกรม PM.EXE

ของพรินต์มาสเตอร์พลัส และใช้โปรแกรมที่เราสร้างขึ้นเองคือโปรแกรม 6.11 ซึ่งอยู่ตอนท้ายของบทนี้

วิธีแยกข้อมูลภาพด้วยโปรแกรม PM.EXE เริ่มโดยการรันโปรแกรม PM.EXE แล้วดำเนินการตามขั้นตอนดังต่อไปนี้

จากเมนูแรก เลือกรายการ Drawing Pad เพื่อเรียกกราฟิกเอดิเตอร์

จากกราฟิกเอดิเตอร์ เลือก L เพื่อขอนำภาพมาที่จอภาพ (load graphic)
เลือกและระบุชื่อตู้ดิสก์ที่เก็บไฟล์ STANDARD.SHP
เลือกชื่อไฟล์ STANDARD
เลือกชื่อภาพที่ต้องการแล้วกดคีย์ Enter
กดคีย์ Ctrl-S เพื่อขอบันทึกภาพไว้ในไฟล์
ป้อนชื่อภาพ
เลิกใช้โปรแกรม

สมมติว่าเราเลือกภาพ GIFT ซึ่งเป็นภาพกล่องของขวัญ ข้อมูลภาพ GIFT จะถูกแยกออกมาเก็บไว้ในไฟล์ CUSTOM.SHP ส่วนชื่อภาพที่เราป้อนเข้าไปจะเก็บอยู่ในไฟล์ CUSTOM.SDR ไฟล์ที่เราต้องการในที่นี้คือไฟล์ CUSTOM.SHP ขั้นต่อไปเราจะต้องเปลี่ยนชื่อไฟล์ CUSTOM.SHP ให้เป็นชื่ออื่นเช่น GIFT.SHP หรือ GIFT.PIC เพื่อป้องกันการนำภาพต่อๆ ไปมารวมเข้าในไฟล์ ทั้งนี้เพราะเมื่อถูกสั่งให้นำภาพ 1 ภาพขึ้นมาแก้ไขและบันทึก พรินต์มาสเตอร์พลัสจะนำภาพนั้นไปบันทึกไว้ในไฟล์ CUSTOM

ข้อมูลในไฟล์ภาพที่แยกออกมาด้วยวิธีการดังกล่าวนี้ เป็นข้อมูลประเภทรหัส ASCII ซึ่งในเทอร์โบปาสคาลกำหนดให้เป็นข้อมูลชนิดไบต์ (byte) มีขนาด 577 ไบต์ต่อไฟล์ ส่วนที่เป็นข้อมูลภาพเริ่มจากไบต์ที่ 5 ถึงไบต์ที่ 576 สำหรับ 4 ไบต์แรกจะเป็นค่าขนาดของภาพและไบต์สุดท้ายเป็นรหัสจบภาพ ดังนั้นข้อมูลภาพจึงมีจำนวนเท่ากับ 572 ไบต์ซึ่งคำนวณได้จาก (576 – 5) + 1 = 572

วิธีแสดงภาพจากไฟล์มีขั้นตอนดังนี้

1. เปิดไฟล์และอ่านข้อมูลทั้งหมดมาเก็บไว้ในอาร์เรย์ของไบต์ อาร์เรย์นี้มีสมาชิก 577 รายการ เรียงตามลำดับตั้งแต่ 1 ถึง 577

2. เปลี่ยนข้อมูลแต่ละไบต์ให้เป็นบิตโดยกำหนดให้ 1 บิตเป็น 1 พิกเซล ซึ่งจะได้ไบต์ละ 8 พิกเซล วิธีการเปลี่ยนไบต์ให้เป็นพิกเซลที่สะดวกที่สุดสำหรับกรณีนี้คือการใช้ไพรซีเยอร์ **SetLineStyle** ซึ่งมีรูปแบบดังนี้

```
SetLineStyle (LineStyle : Word ; Pattern : Word;
              Thickness : Word)
```

โดย LineStyle เป็นลักษณะของเส้นซึ่งกำหนดเป็นคอนสแตนต์ไว้ดังนี้

```
CONST
SolidLn    = 0;  (เส้นทึบล่าง)
DottedLn   = 1;  (เส้นจุด)
CenterLn   = 2;  (เส้นทึบกลาง)
DashedLn   = 3;  (เส้นประ)
UserBitLn  = 4;  (ผู้ใช้กำหนด)
```

Thickness เป็นความหนาของเส้นกำหนดเป็นคอนสแตนต์ไว้ดังนี้

```
CONST
NormWidth   = 1;  (เส้นหนาปกติ)
ThickWidth  = 3;  (เส้นหนา 3 เท่าของปกติ)
```

Pattern เป็นลักษณะของเส้นที่เรากำหนดขึ้นเอง ซึ่งกำหนดได้ครั้งละไม่เกิน 2 ไบต์ (16 พิกเซล)

การใช้ไพรซีเยอร์ SetLineStyle เรากำหนดพารามิเตอร์ต่างๆ ดังนี้

```
SetLineStyle (UserBitLn, ไบต์จากอาร์เรย์, 1)
```

โดยเลือกกำหนดขนาดเส้นให้เป็นเส้นหนาปกติ และกำหนดลักษณะของเส้นด้วยค่าไบต์ที่เก็บอยู่ในแต่ละสมาชิกของอาร์เรย์ ซึ่งจะกำหนดค่าให้ครั้งละ 1 ไบต์ทำให้เกิดเส้นที่มี 8 จุดอยู่ทางขวามือ ส่วน 8 จุดทางซ้ายมือจะว่างไว้

3. ลากเส้นตามลักษณะที่กำหนดไว้ในข้อ 2 ด้วยโพรซีเยอร์ Line โดยกำหนดพารามิเตอร์ดังนี้

Line (X, Y, X + 15, Y) {จอภาพเฮอร์คิวลิส}

ซึ่ง X,Y คือตำแหน่งเริ่มต้นของเส้นตรง

และ X + 15, Y คือตำแหน่งปลายของเส้นตรง

จะเห็นว่าเราต้องลากเส้นตรงให้ยาวเส้นละ 16 จุด

สำหรับจอภาพวีจีเอให้ใช้ X - 15 แทน X + 15

4. กำหนดจำนวนเส้นในแต่ละแถวให้เท่ากับความกว้างของภาพ โดยการลากเส้นต่อเนื่องกัน จนได้จำนวนจุดเท่ากับความกว้างของภาพ ในกรณีนี้ภาพมีความกว้างเป็น 88 พิกเซล ดังนั้นจำนวนเส้นในแต่ละแถวจึงเท่ากับ 88/8 = 11 เส้น ซึ่งเขียนเป็นสเตตเมนต์ได้ดังนี้

```
Line (X, Y, X + 15, Y);
Inc (X, 8); หรือ  X  :=  X + 8;
```

สเตตเมนต์ชุดนี้ต้องทำงานเป็นลูป โดยจะออกจากลูปเมื่อจำนวนเส้นหรือจำนวนไบต์ มีค่ามากกว่า 11 ขอให้สังเกตว่าแม้จะลากเส้นยาว 16 จุดแต่ตำแหน่งเริ่มต้นของเส้นหลังอยู่ห่างจากตำแหน่งเริ่มต้นของเส้นก่อนเพียง 8 จุด เนื่องจาก 8 จุดทางซ้ายมือถูกปล่อยให้ว่างไว้ดังกล่าวมาแล้ว

5. กำหนดจำนวนแถวให้เท่ากับความสูงของภาพ และกำหนดตำแหน่งเริ่มต้นของแถวใหม่ให้ตรงกับแถวก่อน เนื่องจากโปรแกรมต้องทำงานเป็นลูป ดังนั้น เพื่อความสะดวกจึงกำหนดจำนวนแถวด้วยจำนวนไบต์ของข้อมูลที่มีอยู่ แทนการกำหนดจำนวนแถวโดยตรง ซึ่งหมายความว่าให้โปรแกรมลากเส้นจนหมดข้อมูลที่มีอยู่ สำหรับภาพที่เริ่มจากไบต์ที่ 5 และจบที่ไบต์ 576 เขียน สเตตเมนต์ได้ดังนี้

```
FOR ByteNo  :=  5 TO 576 DO
```

จะมีการเริ่มต้นแถวใหม่เมื่อได้ทดสอบเงื่อนไข และพบว่าได้ลากเส้นครบ 1 แถวแล้ว วิธีเริ่มต้นแถวใหม่ทำโดยการเพิ่มค่า Y ขึ้นอีก 1 และกำหนดค่า X ให้เท่ากับค่าตำแหน่งเริ่มต้นของแถวก่อน สำหรับแถวที่มี 11 เส้น (11 ไบต์) เขียนสเตตเมนต์ได้ดังนี้

```
IF ByteInLine  >  11 THEN
BEGIN
   ByteInLine  :=  0;
   Inc (Y) ;
   X  :=  StartX
END;
```

เมื่อดำเนินการตามข้อ 1 ถึง 5 จะมีภาพเกิดขึ้นที่จอภาพ 1 ภาพ ซึ่งเป็นภาพขนาดประมาณ 1 ใน 48 ของจอภาพ

ตัวอย่าง โปรแกรม 6.1 แสดงภาพจากไฟล์ภาพชื่อ BIRD.GTP โปรแกรมจะเริ่มด้วยการอ่านข้อมูลจากไฟล์แล้วแสดงภาพตามข้อมูลนั้นด้วยโพรซีเยอร์ DisplayPicture พารามิเตอร์ของโพรซีเยอร์เป็นตำแหน่ง X,Y ที่มุมบนซ้ายของภาพ โปรแกรมนี้สามารถแสดงภาพจากไฟล์ย่อยที่แยกมาจากไฟล์ STANDARD.SHP ได้ทุกไฟล์ และสามารถแสดงภาพจากไฟล์ชนิด .GTP ที่สร้างขึ้นตามบทที่ 3 ได้ด้วย โดยการเปลี่ยนชื่อไฟล์ในโปรแกรมนี้ให้เป็นตามที่ต้องการ

โปรแกรม 6.1

```
PROGRAM PictureFromPrintMaster;
USES Crt,Graph;
TYPE Pixel      = ARRAY[1..577] OF Byte;
```

```
     PixelRec = RECORD
                    PictureItem : Pixel;
                END;
VAR PixelArray   : Pixel;
    PixelRecord : PixelRec;

PROCEDURE ReadPictureFile;
VAR PictureFile : File OF PixelRec;
BEGIN
   Assign(PictureFile,'BIRD.GTP');
   Reset(PictureFile);
   Read(PictureFile,PixelRecord);
   Close(PictureFile);
   PixelArray := PixelRecord.PictureItem;
END;

PROCEDURE DisplayPicture(X,Y : Integer);
VAR GraphDriver,GraphMode       : Integer;
     ByteNo,ByteInLine,StartX   : Integer;
BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  StartX := X; ByteInLine := 1;
  FOR ByteNo := 5 TO 576 DO
  BEGIN
     SetLineStyle(UserBitLn,PixelArray[ByteNo],1); {thin line}
     IF GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
     Line(X,Y, X+15,Y); Inc(X,8); Inc(ByteInLine);
     IF ByteInLine > 11 THEN  {number of byte in a line}
     BEGIN
```

```
      ByteInLine := 1;        {start new line}
      X := StartX; Inc(Y)
    END;
  END;
END;

BEGIN
  ReadPictureFile;
  DisplayPicture(0,0);
  REPEAT UNTIL KeyPressed;
  CloseGraph;
END.
```

6.1.2 การแสดงภาพจากไฟล์ที่มีหลายภาพ ไฟล์ข้อมูลภาพที่นำมาเป็นตัวอย่างในหัวข้อนี้อาจจะใช้ไฟล์ภาพจากพรินต์มาสเตอร์พลัสคือ ไฟล์ STANDARD.SHP หรือ สร้างไฟล์ขึ้นใหม่แล้วนำข้อมูลภาพที่มีอยู่ในไฟล์ BIRD.GTP และ PUPPY.GTP เข้าไปรวมอยู่ด้วยกันในไฟล์เดียวก็ได้

การนำข้อมูลในไฟล์ BIRD.GTP และ PUPPY.GTP มารวมกันอยู่ในไฟล์ใหม่สามารถทำได้ด้วยโปรแกรมต่อไปนี้

ตัวอย่าง โปรแกรม 6.2 ใช้สำหรับรวมข้อมูลภาพจากไฟล์หลายไฟล์ให้มาเรียงลำดับอยู่ในไฟล์เดียวกัน ในที่นี้ได้ตั้งชื่อไฟล์รวมภาพเป็น MANYPIC.GTP การรวมไฟล์ทำได้ง่ายๆ โดยป้อนชื่อไฟล์ย่อย (เช่น BIRD.GTP) ให้แก่โปรแกรมไปตามลำดับ และจบโปรแกรมโดยกดคีย์ Enter ที่ตำแหน่งแรกของชื่อไฟล์เราสามารถใช้โปรแกรมนี้รวมไฟล์ได้หลายครั้งซึ่งจะมีผลให้ไฟล์ MANYPIC.GTP มีขนาดโตขึ้นเรื่อยๆ ครั้งละ 577 ไบต์

โปรแกรม 6.2

```
PROGRAM AddPictureToFile;
{add a picture to picture library file}
TYPE Pixel     = ARRAY[1..577] OF Byte;
     PixelRec = RECORD
                    PictureItem : Pixel;
                END;
VAR PixelRecord : PixelRec;
    PictureFile : File OF PixelRec;
    FileName    : STRING[14];

PROCEDURE ReadPictureFile;
VAR PictureFile1 : File OF PixelRec;
BEGIN
   Assign(PictureFile1,FileName);
   Reset(PictureFile1);
   Read(PictureFile1,PixelRecord);
   Close(PictureFile1)
END;

BEGIN
  REPEAT
      WriteLn('Add a Picture from 577 byte file to MANYPIC.GTP');
      Write('Enter Name of Picture File : '); ReadLn(FileName);
      IF Length(FileName) > 0 THEN
      BEGIN
          WriteLn('Add ',FileName,
                   ' to MANYPIC.GTP. Please Wait ...');
          Assign(PictureFile,'MANYPIC.GTP');
```

```
        {$I-} Reset(PictureFile); {$I+}
        IF IoResult <> 0 THEN          {MANYPIC.GTP not found}
        BEGIN                          {create file}
          ReWrite(PictureFile);
          Reset(PictureFile)
        END;
        Seek(PictureFile,FileSize(PictureFile));  {seek eof}
        ReadPictureFile;
        Write(PictureFile,PixelRecord);
        Close(PictureFile)
      END;
    UNTIL Length(FileName) < 1
END.
```

ในไฟล์ที่มีหลายภาพ ข้อมูลของแต่ละภาพจะเรียงกันตามลำดับจากภาพแรกถึงภาพสุดท้าย ส่วนใหญ่มักกำหนดให้แต่ละภาพมีจำนวนไบต์เท่ากันเช่นไฟล์ STANDARD.SHP แต่ละภาพมีขนาด 577 ไบต์ ไฟล์ MANYPIC.GTP ก็เช่นเดียวกัน แต่ขณะนี้ไฟล์นี้มีเพียง 2 ภาพเท่านั้น ปัญหาของการแสดงภาพจากไฟล์ที่มีหลายภาพคือ ทำอย่างไรจึงจะสามารถเลือกภาพที่ต้องการมาแสดงได้อย่างรวดเร็วและถูกต้อง โดยเฉพาะจากไฟล์ที่มีภาพรวมกันอยู่เป็นจำนวนมาก เช่น ในไฟล์ STANDARD.SHP ซึ่งมีภาพถึง 122 ภาพ

วิธีเข้าถึงข้อมูลในไฟล์ ณ ตำแหน่งที่ต้องการ สามารถทำได้โดยกำหนดให้ข้อมูลในไฟล์มีชนิดเป็นเรคอร์ด และข้อมูลในเรคอร์ดมีชนิดเป็นอาร์เรย์ของไบต์ ซึ่งมีจำนวนสมาชิกเท่ากับจำนวนไบต์ของ 1 ภาพ ดังตัวอย่างต่อไปนี้

```
  TYPE
    Pixel       =  ARRAY [1..577] OF Byte;
    PixelRec    =  RECORD
```

```
                PictureItem : Pixel;
            END;
    VAR
        PixelArray  :  Pixel;
        PixelRecord :  PixelRec;
```

PixelRecord เป็นเรคอร์ดสำหรับเก็บข้อมูลภาพ 1 ภาพ การอ่านเรคอร์ดที่ต้องการทำได้โดยระบุลำดับที่ของเรคอร์ดนั้น ลำดับที่ของเรคอร์ดในไฟล์เริ่มจาก 0, 1,2 ไปตามลำดับ ดังนั้นถ้าเราต้องการอ่านข้อมูลภาพที่ 1 ก็จะต้องระบุลำดับที่ของเรคอร์ดเป็น 0 เพราะฉะนั้นการระบุค่าลำดับที่ของเรคอร์ดที่ใช้ในโปรมแกรมจึงเป็นดังนี้

ลำดับที่เรคอร์ดที่ต้องการ = ลำดับที่ของภาพ - 1

ถ้ากำหนดให้

```
PictureFile  :  File of PicturRec;
StartPic     :  Integer;
```

โดย StartPic เป็นลำดับที่ของภาพที่ต้องการอ่านจากไฟล์ จะเขียนสเตตเมนต์ได้ดังนี้

```
Seek (PictureFile, StartPic-1);
Read (PictureFile, PixelRecord)
```

เช่น ถ้าต้องการอ่านข้อมูลภาพที่ 5 เราก็กำหนดค่า StartPic ให้เป็น 5 ตามปกติ แต่โปรแกรมจะไปอ่านเรคอร์ดลำดับที่ 4 จากไฟล์มาเก็บไว้ใน PixelRecord ซึ่งเป็นการทำงานที่ถูกต้องตรงตามที่เราต้องการ เมื่อมีข้อมูลภาพอยู่ใน PixelRecord แล้ว เราก็ดำเนินการต่อไปเช่นเดียวกับการแสดงภาพจากไฟล์ที่มี 1 ภาพคือ กำหนดค่าใน PixelRcord ให้แก่ PixelArray แล้วแสดงภาพด้วยโพรซีเยอร์ DisplayPicture ตามที่ได้กล่าวแล้วในข้อ 6.1.1

ตัวอย่าง โปรมแกรม 6.3 แสดงภาพขนาด 88 x 52 พิกเซลจากไฟล์ที่มีหลายภาพซึ่งอาจจะเป็นไฟล์ STANDARD.SHP หรือไฟล์ MANYPIC.GTP ก็ได้ โปรแกรมนี้สามารถแสดงภาพที่จอภาพได้ครั้งละไม่เกิน 49 ภาพ โดยเราจะต้องกำหนดค่า 2 ค่า ให้แก่พารามิเตอร์ของโพรซีเยอร์ ReadStandardShp คือ StartPic เป็นลำดับที่ของภาพเริ่มต้นและ NoOfPic เป็นจำนวนภาพที่ต้องการแสดงใน 1 จอภาพ

โปรแกรม 6.3

```
PROGRAM PictureFromStandardShp;
USES Crt,Graph;
TYPE Pixel      = ARRAY[1..577] OF Byte;
     PixelRec = RECORD
                    PictureItem : Pixel;
                END;
VAR GraphDriver,GraphMode     : Integer;
    PixelArray   : Pixel;
    PixelRecord : PixelRec;

PROCEDURE DisplayPicture(X,Y : Integer);
VAR ByteNo,ByteInLine,StartX   : Integer;
BEGIN
   StartX := X; ByteInLine := 1;
   FOR ByteNo := 5 TO 576 DO
   BEGIN
      SetLineStyle(UserBitLn,PixelArray[ByteNo],1); {thin line}
      If GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
      Line(X,Y, X+15,Y); Inc(X,8); Inc(ByteInLine);
      IF ByteInLine > 11 THEN  {number of byte in a line}
```

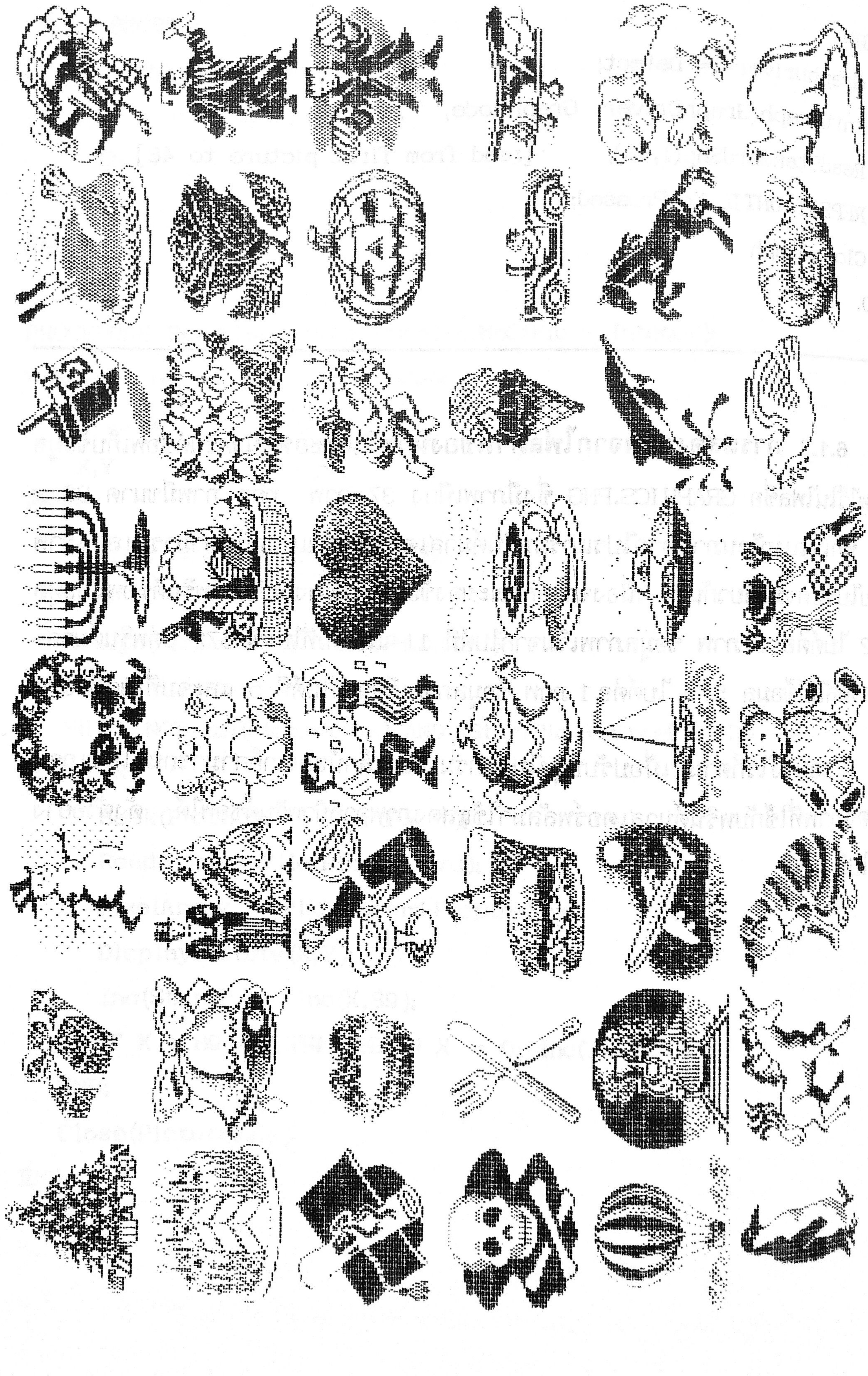
```
      BEGIN
        ByteInLine := 1;
        X := StartX; Inc(Y)
      END
    END
END;

PROCEDURE ReadStandardShp(StartPic,NoOfPic : Integer);
VAR PictureFile : File OF PixelRec;
    StopPic     : Integer;
    X,Y         : Integer;
BEGIN
   StopPic := StartPic + NoOfPic;
   X := 0; Y := 0;
   Assign(PictureFile,'STANDARD.SHP');
   Reset(PictureFile);
   WHILE NOT Eof(PictureFile) AND (StartPic < StopPic) DO
   BEGIN
      Seek(PictureFile,StartPic-1);
      Read(PictureFile,PixelRecord);
      PixelArray := PixelRecord.PictureItem;
      DisplayPicture(X,Y);
      Inc(StartPic); Inc(X,89);
      IF X > 89 * 7 THEN BEGIN X := 0; Inc(Y,55); END;
   END;
   Close(PictureFile)
END;
```

```
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   ReadStandardShp(1,48);      {read from first picture to 48}
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```

6.1.3 การแสดงภาพจากไฟล์ภาพของนิวพรินต์ชอพ นิวพรินต์ชอพเก็บข้อมูลภาพไว้ในไฟล์ชื่อ GRAPHICS.PNG ซึ่งมีภาพเพียง 37 ภาพ ทุกๆ ภาพมีขนาด 88 x 52 พิกเซลเหมือนภาพจากโปรแกรมพรินต์มาสเตอร์พลัส แต่เราไม่สามารถแสดงภาพด้วยโปรแกรมเดียวกันได้ เนื่องจากลักษณะของข้อมูลแตกต่างกันคือนิวพรินต์ชอพมีข้อมูล 572 ไบต์ต่อ 1 ภาพ ข้อมูลภาพเริ่มจากไบต์ที่ 11 และจบที่ไบต์ที่ 572 แต่พรินต์มาส-เตอร์พลัสมีข้อมูล 577 ไบต์ต่อ 1 ภาพ ข้อมูลภาพเริ่มจากไบต์ที่ 5 และจบที่ไบต์ที่ 576

อย่างไรก็ตาม เมื่อปรับปรุงโปรแกรมเพียงเล็กน้อยเราก็สามารถนำโปรแกรมแสดงภาพที่ใช้กับพรินต์มาสเตอร์พลัสมาใช้แสดงภาพจากนิวพรินต์ชอพได้ ดังตัวอย่างต่อไปนี้



รูปที่ 6.1 ภาพบางส่วนจากไฟล์ STANDARD.SIIP

ตัวอย่าง โปรแกรม 6.4 ใช้แสดงภาพขนาด 88 x 52 พิกเซลจากไฟล์ GRAPHICS.PNG ซึ่งเป็นไฟล์ภาพจากโปรแกรมนิวพรินต์ชอพ โปรแกรมนี้ปรับปรุงมาจากโปรแกรม 6.3 สามารถแสดงภาพที่มีอยู่ในไฟล์ทั้ง 37 ภาพพร้อมกัน หรือแสดงภาพที่กำหนดครั้งละภาพก็ได้ โดยการกำหนดค่าหมายเลขภาพ และจำนวนภาพที่ต้องการให้เป็นพารามิเตอร์ของโพรซีเยอร์ ReadStandardShp เช่นเดียวกับโปรแกรม 6.3

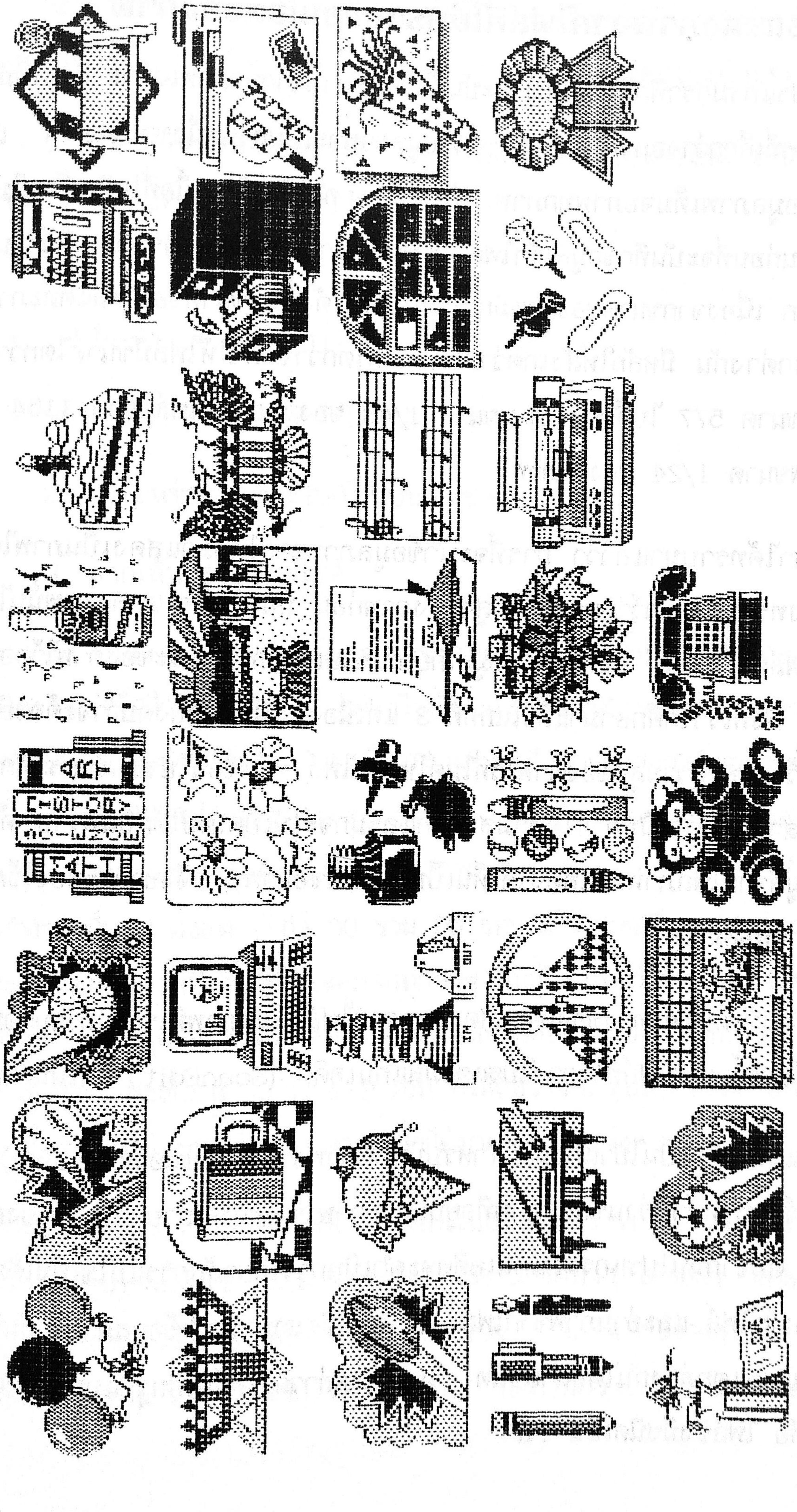
โปรแกรม 6.4

```
PROGRAM PictureFromGraphicsPng;
USES Crt,Graph;
TYPE Pixel    = ARRAY[1..572] OF Byte;
     PixelRec = RECORD
                  PictureItem : Pixel;
                END;
VAR GraphDriver,GraphMode     : Integer;
    PixelArray  : Pixel;
    PixelRecord : PixelRec;

PROCEDURE DisplayPicture(X,Y : Integer);
VAR ByteNo,ByteInLine,StartX  : Integer;
BEGIN
    StartX := X; ByteInLine := 1;
    FOR ByteNo := 11 TO 572 DO
    BEGIN
       SetLineStyle(UserBitLn,PixelArray[ByteNo],1); {thin line}
       IF GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
       Line(X,Y, X+15,Y); Inc(X,8); Inc(ByteInLine);
       IF ByteInLine > 11 THEN  {number of byte in a line}
       BEGIN
          ByteInLine := 1;
          X := StartX; Inc(Y)
```

```
      END
    END
END;

PROCEDURE ReadStandardShp(StartPic,NoOfPic : Integer);
VAR PictureFile : File OF PixelRec;
     StopPic       : Integer;
     X,Y           : Integer;
BEGIN
   StopPic := StartPic + NoOfPic;
   X := 0; Y := 0;
   Assign(PictureFile,'GRAPHICS.PNG');
   Reset(PictureFile);
   WHILE NOT Eof(PictureFile) AND (StartPic < StopPic) DO
   BEGIN
      Seek(PictureFile,StartPic-1);
      Read(PictureFile,PixelRecord);
      PixelArray := PixelRecord.PictureItem;
      DisplayPicture(X,Y);
      Inc(StartPic); Inc(X,89);
      IF X > 89 * 7 THEN BEGIN X := 0; Inc(Y,55); END;
   END;
   Close(PictureFile)
END;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   ReadStandardShp(1,37);      {read from first picture to 37}
   REPEAT UNTIL KeyPressed;
   CloseGraph
END.
```



รูปที่ 6.2 ภาพจากไฟล์ GRAPHICS.PNG

6.2 การแสดงภาพจากไฟล์ที่ไม่ทราบขอบเขตของภาพ

โปรแกรมกราฟิกส์ส่วนใหญ่ จะบันทึกข้อมูลภาพจากจอภาพเฉพาะบริเวณที่มีภาพ สำหรับภาพที่เล็กกว่าจอภาพ ขอบเขตของข้อมูลภาพจะเป็นส่วนหนึ่งของจอภาพ เหตุที่ไม่บันทึกข้อมูลภาพเต็มจอภาพทุกภาพ ก็เพราะจะทำให้เปลืองเนื้อที่ในดิสก์โดยไม่จำเป็น ดังนั้นก่อนที่จะบันทึกข้อมูลเข้าไฟล์ เราจึงต้องกำหนดขอบเขตของภาพไว้ก่อนแล้วจึงสั่งบันทึก เนื่องจากขนาดของภาพมักจะไม่เท่ากัน ดังนั้น ขนาดของไฟล์แต่ละภาพจึงมีขนาดแตกต่างกัน มีหลักให้สังเกตว่าภาพขนาดโตกว่าจะทำให้ไฟล์มีขนาดโตกว่าด้วย เช่น ไฟล์ขนาด 577 ไบต์ใช้เก็บภาพขนาด 1/48 ของจอภาพ ไฟล์ขนาด 1154 ไบต์ใช้เก็บภาพขนาด 1/24 ของจอภาพ

เราได้ทราบมาแล้วว่า การที่จะนำข้อมูลภาพจากไฟล์มาแสดงเป็นภาพได้นั้น เราจะต้องทราบความกว้างและความสูงของภาพก่อน มิฉะนั้นก็จะแสดงภาพนั้นไม่ได้ ตามปกติไฟล์ข้อมูลภาพ มักจะเก็บข้อมูลเกี่ยวกับขอบเขตและลักษณะของภาพไว้ตอนต้นของไฟล์ ดังที่เราได้ศึกษามาแล้วในบทที่ 3 แต่เนื่องจากไม่มีข้อบังคับว่าจะต้องบันทึกค่าความกว้างและความสูงของภาพไว้ที่ไบต์ไหนในไฟล์ ผู้เขียนโปรแกรมกราฟิกส์แต่ละรายจึงสามารถดำเนินการได้ตามสะดวกและมักจะไม่เปิดเผยให้ผู้อื่นทราบ ดังนั้นการนำข้อมูลจากไฟล์ภาพมาแสดงให้เห็นเป็นภาพบนจอภาพ จึงอาจจะต้องใช้ความพยายามมากพอสมควร

เพื่อให้เห็นแนวทางของการแสดงภาพจากไฟล์ข้อมูลภาพที่เราไม่ทราบขอบเขตของภาพ ในที่นี้จะนำไฟล์ที่ได้จากโปรแกรมสแกนเอดิต (ScanEdit) มาเป็นตัวอย่าง

สแกนเอดิตเป็นโปรแกรมใช้สำหรับบันทึกภาพจากสแกนเนอร์(scanner) โดยสแกนเนอร์จะทำหน้าที่อ่านข้อมูลภาพที่อยู่บนกระดาษเข้ามาในหน่วยความจำของคอมพิวเตอร์ ต่อจากนั้นโปรแกรมสแกนเอดิตจะดำเนินการเกี่ยวกับการแก้ไขเพิ่มเติมภาพ บันทึกภาพเข้าไฟล์ และอ่านภาพจากไฟล์ สแกนเอดิตสามารถนำข้อมูลภาพไปบันทึกในไฟล์ที่มีรูปแบบแตกต่างกันได้หลายชนิด ในที่นี้จะกล่าวถึงไฟล์ที่เป็นรูปแบบของสแกนเอดิตเองคือ ไฟล์ซึ่งมีชนิดเป็น TIF

วิธีการแสดงภาพจากไฟล์ชนิด TIF มีขั้นตอนคล้ายกับที่ได้กล่าวถึงแล้วในข้อ 1 ถึง 5 จากหัวข้อ 6.1.1 โดยมีรายละเอียดของการแก้ไขเพิ่มเติมในแต่ละข้อดังต่อไปนี้

1. กำหนดจำนวนสมาชิกของอาร์เรย์ให้เท่ากับขนาดของไฟล์ เช่น ถ้าไฟล์มีขนาด 6852 ไบต์ อาร์เรย์จะเป็นดังนี้

```
CONST
  Pixel  =  ARRAY[1..6852] OF Byte
```

ขนาดของไฟล์ดูได้จากการใช้คำสั่ง DIR ของ DOS

2. เปลี่ยนข้อมูลแต่ละไบต์ให้เป็นพิกเซล

3. ลากเส้นของแต่ละไบต์

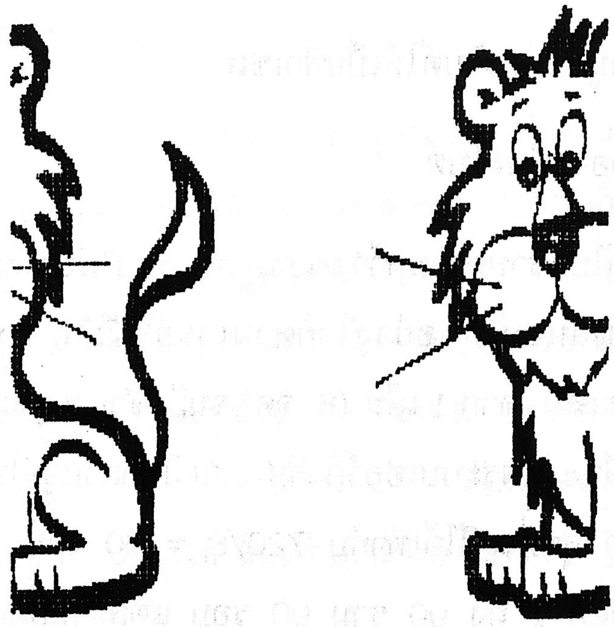
4. เนื่องจากไม่ทราบความกว้างของภาพ จึงไม่สามารถกำหนดจำนวนเส้นในแต่ละแถวให้เป็นที่แน่นอนได้ อย่างไรก็ตามเรายังมีวิธีหาค่านี้ได้ โดยการทดลองเพิ่มจำนวนเส้นขึ้นทีละเส้น จาก 1 เส้น (8 จุด) จนถึงจำนวนสูงสุดที่จอภาพนั้นสามารถทำได้ ตัวอย่าง กรณีจอภาพระบบเฮอร์คิวลีส จะมีจำนวนจุดในแนวนอนสูงสุดเท่ากับ 720 จุด จำนวนเส้นสูงสุดที่จะมีได้เท่ากับ 720/8 = 90 เส้น โปรแกรมในส่วนนี้จึงต้องทำงานเป็นลูป ตั้งแต่ 1 ถึง 90 รวม 90 รอบ แต่ละรอบมีการแสดงภาพ 1 ภาพ เราจะต้องคอยดูภาพที่เกิดขึ้นบนจอภาพทุกภาพ มีหลักสังเกตว่าถ้าจำนวนเส้นไม่ถูกต้องภาพที่เห็นจะเป็นจุดไม่มีระเบียบ เมื่อจำนวนเส้นถูกต้องจะมองเห็นเป็นภาพ แต่อาจจะเป็นภาพที่ไม่สมบูรณ์เช่น ไม่อยู่กลางกรอบ ก็ให้ถือว่าจำนวนเส้นในแต่ละแถวเท่ากับค่านั้น ส่วนความไม่สมบูรณ์ของภาพจะแก้ไขได้ตามวิธีการในข้อ 6

เนื่องจากการทดลองหาจำนวนเส้นในแต่ละแถว เป็นงานที่ต้องใช้เวลามาก ถ้าเราเคยเห็นภาพนั้นมาก่อน อาจจะช่วยลดเวลาลงได้บ้าง โดยการกำหนดค่าเริ่มต้นของลูปให้มากขึ้น เช่น ถ้าเห็นภาพกว้างประมาณ 1 ใน 4 ของจอภาพ ค่าเริ่มต้นของลูปควรจะเป็น 22 (ได้จาก 720/8/4) ถ้าภาพกว้างประมาณครึ่งจอภาพ ค่า

เริ่มต้นควรจะเป็น 45 (ได้จาก 720/8/2) การทำเช่นนี้มีผลให้ไม่ต้องเสียเวลาแสดงภาพในช่วงต้นของลูป ซึ่งเราแน่ใจแล้วว่าไม่เกิดภาพที่ถูกต้องในช่วงนั้น

5. กำหนดจำนวนแถว หรือความสูงของภาพให้เท่ากับจำนวนไบต์ที่มีในไฟล์ตามวิธีที่กล่าวแล้ว

6. กรณีที่ภาพไม่สมบูรณ์ เช่น มีส่วนเกินที่ด้านบนของภาพ ภาพแยกออกเป็น 2 ส่วน ตามตัวอย่างดังแสดงในรูปที่ 6.3



รูปที่ 6.3 ความไม่สมบูรณ์ของภาพ

ความผิดพลาดเหล่านี้เกิดจากการกำหนดไบต์เริ่มต้นของภาพไม่ถูกต้อง ตามปกติเรามักจะเริ่มต้นให้แสดงภาพตั้งแต่ไบต์แรกของไฟล์ ซึ่งมักจะไม่ถูกต้อง เพราะส่วนนั้นเป็นข้อมูลเกี่ยวกับภาพ ไม่ใช่ข้อมูลภาพ ดังนั้นจึงจำเป็นที่เราจะต้องทำการทดลองเพื่อหาค่าดังกล่าวนี้ด้วย วิธีทดลองทำโดยการเปลี่ยนตำแหน่งไบต์เริ่มต้นของภาพไปตามลำดับ จนกระทั่งได้ภาพที่สมบูรณ์ (ถ้าเคยเห็นภาพนั้นมาก่อน) หรือได้ภาพตรงกับความต้องการ (ไม่ว่าจะเคยเห็นภาพนั้นมาก่อนหรือไม่ก็ตาม) ซึ่งโปรแกรมในส่วนนี้จะต้องทำงานเป็นลูป โดยกำหนดค่าเริ่มต้นตั้งแต่ไบต์แรกคือ 1 จนถึงไบต์ที่คาด

ว่าภาพน่าจะสมบูรณ์ ซึ่งอาจจะเป็น 5, 11 หรือค่าอื่นๆ สำหรับไฟล์ชนิด TIF ค่าที่ใช้คือ 343 ซึ่งนับว่าเป็นค่าที่สูงมาก

เมื่อได้ค่าไบต์เริ่มต้นของภาพ (ตามข้อ 6) และค่าจำนวนเส้นในแต่ละแถว (ตามข้อ 4) เป็นที่แน่นอนแล้ว เราก็สามารถกำหนดค่าให้แก่โปรแกรมเพื่อดำเนินการต่างๆ กับภาพนั้นต่อไปได้ เช่น การแสดงภาพบนจอภาพ การพิมพ์ภาพ การบันทึกข้อมูลภาพในไฟล์ตามรูปแบบไฟล์ของเทอร์โบปาสคาล ซึ่งสามารถแสดงภาพได้รวดเร็วกว่า

เพื่อให้เข้าใจเรื่องนี้ได้ชัดเจนขึ้น จะขอนำเอาไฟล์ภาพชนิด TIF ซึ่งมีอยู่ในดิสก์โปรแกรมสแกนเอดิต และไฟล์บางไฟล์ที่สแกนเก็บเอาไว้ มาเป็นตัวอย่างประกอบการอธิบาย ไฟล์ที่จะกล่าวถึงมีดังต่อไปนี้

ชื่อไฟล์	ขนาดไฟล์(ไบต์)	ภาพที่เห็น
LION.TIF	6852	สิงโต
DUCK.TIF	7602	เป็ด
SCORPIO.TIF	3382	แมงป่อง
FLOWER.TIF	5022	ดอกไม้
BABY1.TIF	13122	เด็กชายวรรนุ 3 ขวบ
WANNU1.TIF	29217	เด็กชายวรรนุ 8 ขวบ

จะเห็นได้ว่าไฟล์เหล่านี้มีขนาดแตกต่างกันซึ่งแสดงว่าภาพจากแต่ละไฟล์มีขนาดไม่เท่ากัน เพื่อให้สามารถมองเห็นภาพจากไฟล์ซึ่งเราไม่ทราบรายละเอียดเกี่ยวกับการบันทึกข้อมูล เราจะเขียนโปรแกรมสำหรับทดสอบภาพเหล่านี้ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 6.5 ใช้ทดสอบการแสดงภาพจากไฟล์ข้อมูลภาพที่เราไม่ทราบขอบเขตและไบต์เริ่มต้นของข้อมูลภาพ เราต้องกำหนดค่าต่างๆที่จะใช้ในการทดสอบให้แก่โปรแกรมดังต่อไปนี้

ชื่อไฟล์ข้อมูลภาพ FileName เช่น 'LION.TIF'

ขนาดของไฟล์ FileSize เช่น 6852

ไบต์แรกของข้อมูลภาพน่าจะเริ่มที่... TestPicByteFrom เช่น 342

ไบต์แรกของข้อมูลภาพน่าจะจบที่... TestPicByteTo เช่น 343

ไบต์ใน 1 แถวน่าจะเริ่มด้วยค่า... TestNumOfByteFrom เช่น 30

ไบต์ใน 1 แถวน่าจะจบด้วยค่า... TestNumOfByteTo เช่น 31

โปรแกรม 6.5

```
PROGRAM PictureFromTifFile;
USES Crt,Graph;

(*          specify items for testing hear               *)
(*****************************************************)
CONST FileName            = 'LION.TIF';
      FileSize            = 6852;
      TestPicByteFrom     = 342;
      TestPicByteTo       = 343;
      TestNumOfByteFrom   = 30;
      TestNumOfByteTo     = 31;
(*****************************************************)

TYPE Pixel      = ARRAY[1..FileSize] OF Byte;
     PixelRec = RECORD
                   PictureItem : Pixel;
                END;
VAR PixelArray   : Pixel;
    PixelRecord : PixelRec;

PROCEDURE ReadPictureFile;
VAR PictureFile : File OF PixelRec;
BEGIN
```

```
  Assign(PictureFile,FileName);
  Reset(PictureFile);
  Read(PictureFile,PixelRecord);
  Close(PictureFile);
  PixelArray := PixelRecord.PictureItem;
END;

PROCEDURE DisplayPicture(X,Y : Integer);
VAR GraphDriver,GraphMode      : Integer;
    ByteNo,ByteInLine,StartX   : Integer;
    StartByte,NumberOfByte     : Integer;
    MsgByteNo,MsgStartByte     : STRING[4];
    Ch : Char;
BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  FOR NumberOfByte := TestNumOfByteFrom TO TestNumOfByteTo DO
  BEGIN
     Str(NumberOfByte,MsgByteNo);
  FOR StartByte := TestPicByteFrom TO TestPicByteTo DO
  BEGIN
     Str(StartByte,MsgStartByte);
     ClearDevice;
     X := 0;           Y := 0;
     StartX := X;      ByteInLine := 0;
     FOR ByteNo := StartByte TO FileSize DO
     BEGIN
       SetLineStyle(UserBitLn,PixelArray[ByteNo],1);
       IF GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
       Line(X,Y, X+15,Y);   Inc(X,8);    Inc(ByteInLine);
```

```
        IF ByteInLine = NumberOfByte THEN  {number of byte in a line}
        BEGIN
          ByteInLine := 0;
          Inc(Y);     X := StartX
        END;
      END;
      OutTextXY(NumberOfByte*8+8+12, 0, 'ByteInLine   '+MsgByteNo);
      OutTextXY(NumberOfByte*8+8+12,10, 'StartAtByte '+MsgStartByte);
      OutTextXY(NumberOfByte*8+8+12,20, 'ENTER or ESC');
      Ch := ReadKey; IF Ch = #27 THEN BEGIN CloseGraph; Halt END;
    END; END;
END;

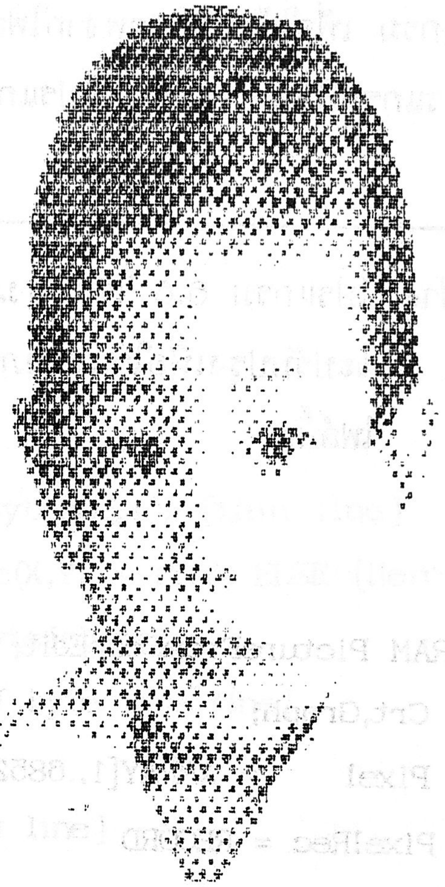
BEGIN
  ReadPictureFile;
  DisplayPicture(0,0);
  CloseGraph;
END.
```

เมื่อใช้โปรแกรม 6.5 ทดสอบไฟล์ข้อมูลภาพชุดนี้ปรากฏว่าได้ผลการทดสอบดังต่อไปนี้

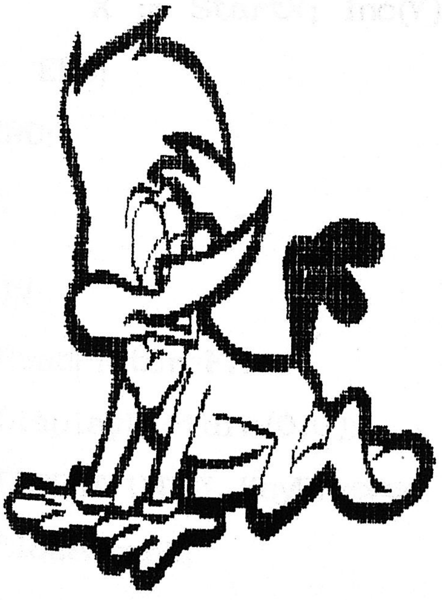
ชื่อไฟล์	ไบต์เริ่มข้อมูลภาพ	ไบต์จบข้อมูลภาพ	จำนวนไบต์ใน 1 แถว
LION.TIF	343	6852	31
DUCK.TIF	343	7602	33
SCORPIO.TIF	343	3382	19
FLOWER.TIF	343	5022	26
BABY1.TIF	343	13122	36
WANNU1.TIF	343	29217	55



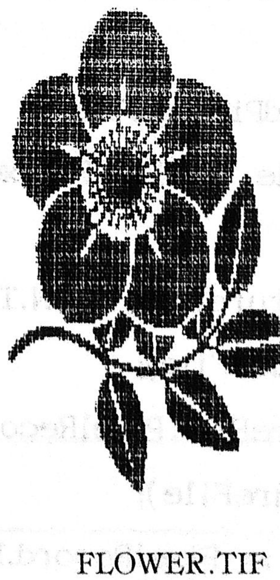
LION.TIF



BABV1.TIF



DUCK.TIF



FLOWER.TIF

รูปที่ 6.4 ภาพที่ได้จากไฟล์ชนิด TIF

เมื่อทราบค่าที่ได้จากการทดสอบแล้ว เราอาจจะนำค่าเหล่านั้นมาเขียนเป็นโปรแกรม เพื่อใช้แสดงภาพจากไฟล์ที่ต้องการเพียงไฟล์เดียวโดยเฉพาะก็ได้ ซึ่งจะได้โปรแกรมที่มีขนาดเล็กกว่าโปรแกรมที่ใช้สำหรับการทดสอบ ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 6.6 ใช้แสดงภาพจากไฟล์ LION.TIF โดยเฉพาะ แต่เราอาจจะปรับปรุงเพื่อใช้แสดงภาพจากไฟล์อื่นได้ โดยการเปลี่ยนข้อมูลให้เป็นของไฟล์นั้น

โปรแกรม 6.6

```
PROGRAM PictureFromScanEdit;
USES Crt,Graph;
TYPE Pixel     = ARRAY[1..6852] OF Byte;
    PixelRec = RECORD
                 PictureItem : Pixel;
               END;
VAR PixelArray  : Pixel;
    PixelRecord : PixelRec;

PROCEDURE ReadPictureFile;
VAR PictureFile : File OF PixelRec;
BEGIN
   Assign(PictureFile,'LION.TIF');
   Reset(PictureFile);
   Read(PictureFile,PixelRecord);
   Close(PictureFile);
   PixelArray := PixelRecord.PictureItem;
END;
```

```
PROCEDURE DisplayPicture(X,Y : Integer);
VAR GraphDriver,GraphMode       : Integer;
   ByteNo,ByteInLine,StartX    : Integer;
BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  StartX := X; ByteInLine := 1;
  FOR ByteNo := 343 TO 6852 DO
  BEGIN
    SetLineStyle(UserBitLn,PixelArray[ByteNo],1); {thin line}
    IF GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
    Line(X,Y, X+15,Y); Inc(X,8); Inc(ByteInLine);
    IF ByteInLine > 31 THEN  {number of byte in a line}
    BEGIN
      ByteInLine := 1;        {start new line}
      X := StartX; Inc(Y)
    END;
  END;
END;

BEGIN
  ReadPictureFile;
  DisplayPicture(0,0);
  REPEAT UNTIL KeyPressed;
  CloseGraph;
END.
```

ตัวอย่าง มีโปรแกรมสำเร็จรูปเกี่ยวกับกราฟิกส์โปรแกรมหนึ่งซึ่งคนไทยเป็นผู้สร้าง ใช้ชื่อโปรแกรมว่า สระบุรี ในดิสก์ของโปรแกรมนี้มีไฟล์ชื่อ LADY.PIC ซึ่งเป็น

ไฟล์ภาพหญิงสาวสวมแว่นตา ภาพมีขนาดประมาณ 3/4 ของจอภาพ ไฟล์มีขนาด 18406 ไบต์ เมื่อเราใช้โปรแกรม 6.5 ทดสอบแล้ว จะได้ค่าต่างๆ เกี่ยวกับภาพ ซึ่งสามารถนำมาเขียนเป็นโปรแกรม สำหรับแสดงภาพนี้โดยเฉพาะ ดังแสดงในโปรแกรม 6.7 ต่อไปนี้

โปรแกรม 6.7

```
PROGRAM PictureFromLadyPic;
USES Crt,Graph;
TYPE Pixel    = ARRAY[1..18406] OF Byte;
     PixelRec = RECORD
                  PictureItem : Pixel;
                END;
VAR PixelArray   : Pixel;
    PixelRecord : PixelRec;

PROCEDURE ReadPictureFile;
VAR PictureFile : File OF PixelRec;
BEGIN
   Assign(PictureFile,'LADY.PIC');
   Reset(PictureFile);
   Read(PictureFile,PixelRecord);
   Close(PictureFile);
   PixelArray := PixelRecord.PictureItem;
END;

PROCEDURE DisplayPicture(X,Y : Integer);
VAR GraphDriver,GraphMode        : Integer;
    ByteNo,ByteInLine,StartX     : Integer;
BEGIN
```

```
    GraphDriver := Detect;
    InitGraph(GraphDriver, GraphMode, '');
    StartX := X; ByteInLine := 0;
    FOR ByteNo := 12 TO 18406 DO
    BEGIN
      SetLineStyle(UserBitLn,PixelArray[ByteNo],1); {thin line}
      IF GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
      Line(X,Y, X+15,Y); Inc(X,8); Inc(ByteInLine);
      IF ByteInLine = 65 THEN  {number of byte in a line}
      BEGIN
        ByteInLine := 0;
        X := StartX; Inc(Y)
      END;
    END;
  END;

BEGIN
   ReadPictureFile;
   DisplayPicture(0,0);
   REPEAT UNTIL KeyPressed;
   CloseGraph;
END.
```

จากวิธีการและตัวอย่างที่ผ่านมา ช่วยให้เราได้แนวความคิดเกี่ยวกับวิธีดำเนินการกับไฟล์ภาพจากโปรแกรมสำเร็จรูป ทั้งประเภทที่ทราบรายละเอียดของภาพ และประเภทที่ไม่ทราบรายละเอียดของภาพในไฟล์ อย่างไรก็ตามใคร่ขอย้ำว่าการบันทึกข้อมูลภาพมีวิธีการอีกมากมายหลายอย่าง ซึ่งแล้วแต่ว่าผู้เขียนโปรแกรมจะคิดหรือเลือกใช้วิธีการใด



รูปที่ 6.5 ภาพจากไฟล์ LADY.PIC

6.3 วิธีเปลี่ยนไฟล์ข้อมูลภาพจากโปรแกรมอื่นให้เป็นไฟล์ข้อมูลภาพแบบเทอร์โบปาสคาล

ดังที่เราได้ทราบมาแล้วว่า การบันทึกข้อมูลภาพในไฟล์ตามแบบของเทอร์โบปาสคาลจะสามารถแสดงภาพได้รวดเร็วกว่า ด้วยโปรแกรมขนาดเล็กกว่า ดังนั้นเราจึงควรดัดแปลงไฟล์ภาพจากโปรแกรมอื่น มาเป็นไฟล์ตามแบบของเทอร์โบปาสคาล ซึ่งเราสามารถทำได้ 2 วิธีการคือ การบันทึกข้อมูลจากจอภาพ และการเปลี่ยนข้อมูลในไฟล์ แต่ละวิธีการมีรายละเอียดดังต่อไปนี้

6.3.1 การบันทึกข้อมูลจากจอภาพ เราได้เคยศึกษาวิธีการบันทึกภาพวิธีนี้มาแล้วในบทที่ 3 โดยเรามีโปรแกรม 3.7 ทำหน้าที่นำข้อมูลภาพจากจอภาพมาบันทึกไว้ในไฟล์ตามรูปแบบของเทอร์โบปาสคาล การบันทึกภาพที่จะกล่าวถึงในหัวข้อนี้ก็มีหลักการเช่นเดียวกันคือ นำข้อมูลภาพจากไฟล์มาแสดงเป็นภาพที่จอภาพ แล้วให้โปรแกรมนำข้อมูลภาพในบริเวณที่ต้องการมาบันทึกไว้ในไฟล์ เพียงแต่ในหัวข้อนี้ได้มีการปรับปรุงโปรแกรมให้สามารถทำงานได้กว้างขวางมากกว่าเดิมคือ สามารถใช้ได้กับไฟล์หลายขนาดและหลายลักษณะ ดังตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 6.8 ใช้แสดงภาพจากไฟล์ภาพและนำข้อมูลภาพจากจอภาพมาบันทึกไว้ในไฟล์ตามแบบไฟล์ภาพของเทอร์โบปาสคาล การใช้โปรแกรมกับไฟล์ภาพแต่ละไฟล์เราจะต้องระบุค่าต่างๆให้แก่โปรแกรมดังตัวอย่างต่อไปนี้

ชื่อไฟล์ภาพเดิม	PictureFileName	เช่น	'LADY.PIC'
ชื่อไฟล์เก็บภาพใหม่	GtpFileName	เช่น	'LADY.GTP'
ขนาดของไฟล์เป็นไบต์	FileSize	เช่น	18406
ไบต์เริ่มต้นข้อมูลภาพ	StartByte	เช่น	12
ความกว้างของภาพเป็นไบต์	NoOfByteInLine	เช่น	65

โปรแกรม 6.8

```
PROGRAM PictureToGtp;
{save picture on screen in Turbo Pascal Graphic format GTP}
USES Crt,Graph;

(*     CONST must be change for another graphic file        *)
(**********************************************************)
CONST PictureFileName = 'LADY.PIC'; {original file name}
      GtpFileName     = 'LADY.GTP'; {new GTP file name}
      FileSize        = 18406; {file size in byte}
      StartByte       = 12;     {first byte of picture data}
      NoOfByteInLine  = 65;     {number of byte in a line}

(**********************************************************)

TYPE Pixel      = ARRAY[1..FileSize] OF Byte;
   PixelRec = RECORD
                PictureItem : Pixel;
              END;
```

```
VAR  PixelArray      : Pixel;
     PixelRecord     : PixelRec;
     X1,Y1, X2,Y2    : Integer;
     PictureFile     : File;
     ImagePointer    : Pointer;
     MemUsed         : Word;    {byte of memory used in the picture}
     Ch              : Char;

PROCEDURE ReadPictureFile;
VAR  PictureFile : File OF PixelRec;
BEGIN
   Assign(PictureFile,PictureFileName);
   Reset(PictureFile);
   Read(PictureFile,PixelRecord);
   Close(PictureFile);
   PixelArray := PixelRecord.PictureItem;
END;

PROCEDURE DisplayPicture(X,Y : Integer);
VAR GraphDriver,GraphMode        : Integer;
    ByteNo,ByteInLine,StartX     : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   StartX := X; ByteInLine := 0;
   FOR ByteNo := StartByte TO FileSize DO
   BEGIN
      SetLineStyle(UserBitLn,PixelArray[ByteNo],1); {thin line}
      If GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
      Line(X,Y, X+15,Y); Inc(X,8); Inc(ByteInLine);
```

```
      IF ByteInLine = NoOfByteInLine THEN  {number of byte in a line}
      BEGIN
        ByteInLine := 0;
        X := StartX; Inc(Y)
      END;
    END;
END;

BEGIN
  X1 := 0; Y1 := 0;
  ReadPictureFile;
  DisplayPicture(X1, Y1);
  Ch := ReadKey;
  IF Ch = #27 THEN Halt;
  X1 := X1 + 8;
  X2 := NoOfByteInLine * 8 + X1 - 1;
  Y2 := Round((FileSize - StartByte + 1) / NoOfByteInLine) + 1;
  MemUsed := ImageSize(X1, Y1, X2, Y2);
  GetMem(ImagePointer,MemUsed);
  GetImage(X1, Y1, X2, Y2, ImagePointer^);
  WHILE MemUsed MOD 128 <> 0 DO MemUsed := MemUsed + 1;
  MemUsed := MemUsed DIV 128; {number of record}
  Assign(PictureFile,GtpFileName);
  ReWrite(PictureFile);
  BlockWrite(PictureFile,ImagePointer^,MemUsed);
  Close(PictureFile);
  CloseGraph
END.
```

6.3.2 การเปลี่ยนข้อมูลในไฟล์ให้เป็นแบบเทอร์โบปาสคาล การเปลี่ยนแปลงข้อมูลตามวิธีการนี้เป็นการแก้ไขที่ตัวข้อมูลซึ่งอยู่ในไฟล์โดยตรง ไม่จำเป็นต้องเกี่ยวข้องกับการแสดงภาพที่จอภาพแต่อย่างใด ข้อมูลเกี่ยวกับไฟล์ที่ต้องใช้สำหรับการเปลี่ยนข้อมูลภาพมีดังนี้

ชื่อไฟล์ภาพเดิม	PicFileName	เช่น	'LION.TIF'
ขนาดของไฟล์เป็นไบต์	PicFileSize	เช่น	6852
ชื่อไฟล์ภาพใหม่	GtpFileName	เช่น	'LION.GTP'
ไบต์เริ่มต้นของข้อมูลภาพ	StartPicByte	เช่น	343
ไบต์สิ้นสุดของข้อมูลภาพ	StopPicByte	เช่น	6852
ความกว้างของภาพเป็นไบต์	ByteOfPicWidth	เช่น	31

ลำดับขั้นการเปลี่ยนข้อมูลในไฟล์มีดังนี้

1. กำหนดชนิดข้อมูลเป็นเรคอร์ดของอาร์เรย์ซึ่งมีสมาชิกเท่ากับขนาดของไฟล์
2. อ่านข้อมูลจากไฟล์เดิมเข้ามาเก็บในอาร์เรย์
3. เปลี่ยนแปลงข้อมูลในอาร์เรย์ให้มีรูปแบบตามเงื่อนไขของเทอร์โบปาสคาล ด้วยวิธีการดังต่อไปนี้

ก. เก็บค่าความกว้างของภาพไว้ที่ไบต์ 1 และ 2 ตามเงื่อนไขต่อไปนี้

ถ้าความกว้างของภาพ <= 256

ให้ไบต์ 1 = ความกว้างของภาพ - 1

ให้ไบต์ 2 = 0

ถ้าความกว้างของภาพ > 256

ให้ไบต์ 1 = (ความกว้างของภาพ MOD 256) - 1

ให้ไบต์ 2 = ความกว้างของภาพ DIV 256

ข. คำนวณค่าความสูงของภาพได้ดังนี้

ความสูง = (ไบต์จบภาพ - ไบต์เริ่มภาพ + 1) DIV ความกว้าง

เก็บค่าความสูงของภาพไว้ที่ไบต์ 3 และ 4 ตามเงื่อนไขต่อไปนี้

ถ้าความสูงของภาพ <= 256

ให้ไบต์ 3 = ความสูงของภาพ - 1

ให้ไบต์ 4 = 0

ถ้าความสูงของภาพ > 256

ให้ไบต์ 3 = (ความสูงของภาพ MOD 256) - 1

ให้ไบต์ 4 = ความสูงของภาพ DIV 256

4. เลื่อนข้อมูลภาพจากตำแหน่งเริ่มต้นเดิม มาอยู่ ณ ไบต์ที่ 5 ของไฟล์ใหม่ และเลื่อนข้อมูลในลำดับถัดไปมาเรียงต่อกันตามลำดับ จนถึงไบต์สุดท้ายของข้อมูลภาพ

5. ตั้งแต่ไบต์สุดท้ายของข้อมูลภาพไปจนจบไฟล์ให้เปลี่ยนข้อมูลเป็น 0

6. บันทึกข้อมูลในอาร์เรย์เข้าในไฟล์ใหม่

ตัวอย่าง โปรแกรม 6.9 ทำหน้าที่เปลี่ยนข้อมูลจากไฟล์ภาพของโปรแกรมอื่นให้เป็นไฟล์แบบเทอร์โบปาสคาล เพื่อให้สามารถแสดงภาพได้เร็วขึ้น ตามตัวอย่างนี้เป็นการเปลี่ยนไฟล์ LION.TIF ให้เป็น LION.GTP

โปรแกรม 6.9

```
PROGRAM ConvertPictureToGtpFile;

(*              specify picture details hear              *)
(*********************************************************)
CONST PicFileName      = 'LION.TIF';
      PicFileSize      = 6852;
      GtpFileName      = 'LION.GTP';
      StartPicByte     = 6852;
      StopPicByte      = 343;
```

```
    ByteOfPicWidth   = 31;
(*************************************************)

TYPE Pixel      = ARRAY[1..PicFileSize] OF Byte;
     PixelRec = RECORD
                  PictureItem : Pixel;
                END;
VAR PixelArray   : Pixel;
    PixelRecord : PixelRec;

PROCEDURE ReadPictureFile;
VAR PictureFile : File OF PixelRec;
BEGIN
   Assign(PictureFile,PicFileName);
   Reset(PictureFile);
   Read(PictureFile,PixelRecord);
   Close(PictureFile);
   PixelArray := PixelRecord.PictureItem;
END;

PROCEDURE ConvertToGtpFile;
VAR PictureFile : File OF PixelRec;
    PicWidth,PicHeight,Byte1,Byte2,Byte3,Byte4,ByteNo,
    TpByte : Integer;
BEGIN
   {calculate picture size}
   PicWidth := ByteOfPicWidth * 8;
   IF PicWidth <= 256 THEN
   BEGIN
      Byte1 := PicWidth - 1;
      Byte2 := 0;
```

```
END
ELSE
BEGIN
   Byte2 := PicWidth DIV 256;
   Byte1 := PicWidth MOD 256 - 1
END;
PicHeight := (StopPicByte - StartPicByte + 1)
             DIV ByteOfPicWidth;
IF PicHeight <= 256 THEN
BEGIN
   Byte3 := PicHeight - 1;
   Byte4 := 0;
END
ELSE
BEGIN
   Byte4 := PicHeight DIV 256;
   Byte3 := PicHeight MOD 256 - 1
END;
{transform picture data}
PixelArray[1] := Ord(Chr(Byte1));
PixelArray[2] := Ord(Chr(Byte2));
PixelArray[3] := Ord(Chr(Byte3));
PixelArray[4] := Ord(Chr(Byte4));
TpByte := 5; {start graphic data of TurboPascal}
FOR ByteNo := StartPicByte TO StopPicByte DO
BEGIN
   PixelArray[TpByte] := PixelArray[ByteNo];
   Inc(TpByte);
END;
```

```
  {write file}
  PixelRecord.PictureItem := PixelArray;
  Assign(PictureFile,GtpFileName);
  ReWrite(PictureFile);
  Write(PictureFile,PixelRecord);
  Close(PictureFile);
END;

BEGIN
  ReadPictureFile;
  ConvertToGtpFile;
END.
```

6.4 การแสดงภาพจากไฟล์ภาพของเทอร์โบปาสคาล

ในบทที่ 3 เรามีโปรแกรมเบื้องต้นสำหรับใช้แสดงภาพจากไฟล์ภาพ ซึ่งบันทึกไว้ตามแบบของเทอร์โบปาสคาล เนื่องจากโปรแกรมดังกล่าว (โปรแกรม 3.8) สามารถใช้แสดงภาพจากไฟล์แบบเทอร์โบปาสคาลคือไฟล์ .GTP ได้ทุกไฟล์ ดังนั้นในหัวข้อนี้จึงได้นำโปรแกรม 3.8 มาปรับปรุงเป็นโปรแกรม 6.10 เพื่อให้สะดวกต่อการใช้งานมากยิ่งขึ้น

ตัวอย่าง โปรแกรม 6.10 ใช้สำหรับแสดงภาพจากไฟล์ภาพของเทอร์โบปาสคาล โปรแกรมจะเริ่มด้วยการให้ป้อนชื่อไฟล์ ต่อจากนั้นโปรแกรมจะนำข้อมูลจากไฟล์มาแสดงที่จอภาพอย่างรวดเร็ว

โปรแกรม 6.10

```
PROGRAM PicturesFromGtpFile;
USES Graph,GrDriver;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;
     FileName                : STRING[50];
BEGIN
  Write('Enter Name of Turbo Pascal Graphic File ');
  ReadLn(FileName);
  IF Length(FileName) > 0 THEN
  BEGIN
    Assign(PictureFile,FileName);
    Reset(PictureFile);
    GetMem(PicturePointer,FileSize(PictureFile));
    BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
    Close(PictureFile);
    GraphDriver := Detect;
    InitGraph(GraphDriver,GraphMode,'');
    PutImage(0,0,PicturePointer^,NormalPut);
    Readln
  END;
  CloseGraph
END.
```

6.5 โปรแกรมแยกข้อมูลภาพจากไฟล์ STANDARD.SHP

เพื่อให้เห็นตัวอย่างของการแยกข้อมูลจากไฟล์ใหญ่มาเก็บในไฟล์ที่เล็กกว่า ในที่นี้จึงนำโปรแกรมตัวอย่างซึ่งทำหน้าที่แยกภาพ 1 ภาพจากไฟล์ STANDARD.SHP มาเก็บไว้ในไฟล์ใหม่ซึ่งมีอยู่เพียงภาพเดียว เป็นไฟล์ที่มีขนาด 577 ไบต์

ตัวอย่าง โปรแกรม 6.11 จะเริ่มโดยการถามหมายเลขภาพที่เราต้องการแยกออกมา แล้วโปรแกรมจะแสดงภาพนั้น ถ้าเป็นภาพตรงกับที่เราต้องการให้กดคีย์ S โปรแกรมจะจัดการสร้างไฟล์และบันทึกภาพนั้น ถ้ากดคีย์อื่นจะไม่มีการสร้างไฟล์

โปรแกรม 6.11

```
PROGRAM Convert1PictureFromStandardShp2Gtp;
USES Crt,Graph;
TYPE Pixel     = ARRAY[1..577] OF Byte;
     PixelRec = RECORD
                    PictureItem : Pixel;
                END;
VAR GraphDriver,GraphMode       : Integer;
    PixelArray   : Pixel;
    PixelRecord  : PixelRec;
    PicNo        : Integer;
    FileName     : STRING[3];
    Ch           : Char;

PROCEDURE DisplayPicture(X,Y : Integer);
VAR ByteNo,ByteInLine,StartX   : Integer;
BEGIN
```

```
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   StartX := X; ByteInLine := 1;
   FOR ByteNo := 5 TO 576 DO
   BEGIN
      SetLineStyle(UserBitLn,PixelArray[ByteNo],1); {thin line}
      IF GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
      Line(X,Y, X+15,Y); Inc(X,8); Inc(ByteInLine);
      IF ByteInLine > 11 THEN  {number of byte in a line}
      BEGIN
         ByteInLine := 1;
         X := StartX; Inc(Y)
      END
   END
END;

PROCEDURE ReadStandardShp(PictureNo : Integer);
VAR PictureFile : File OF PixelRec;
BEGIN
   Assign(PictureFile,'STANDARD.SHP');
   Reset(PictureFile);
   Seek(PictureFile,PictureNo-1);
   Read(PictureFile,PixelRecord);
   PixelArray := PixelRecord.PictureItem;
   Close(PictureFile);
   DisplayPicture(0,0)
END;

PROCEDURE ConvertToGtpFile(GtpFileName : STRING);
VAR PictureFile : File OF PixelRec;
```

```
BEGIN
   {transform picture data}
   PixelArray[1] := Ord(Chr(87));
   PixelArray[2] := Ord(Chr(0));
   PixelArray[3] := Ord(Chr(51));
   PixelArray[4] := Ord(Chr(0));
   {write file}
   PixelRecord.PictureItem := PixelArray;
   Assign(PictureFile,GtpFileName);
   ReWrite(PictureFile);
   Write(PictureFile,PixelRecord);
   Close(PictureFile);
END;

BEGIN
   ClrScr;
   WriteLn('Get 1 picture from STANDARD.SHP then save as GTP file');
   Write('Enter number of picture (1.. 122) '); ReadLn(PicNo);
   Str(PicNo,FileName);
   ReadStandardShp(PicNo);
   OutTextXY(0,60,'Press S to SAVE or any key to Quit');
   Ch := ReadKey;
   IF Ch IN['s','S'] THEN ConvertToGtpFile(FileName+'.GTP');
   CloseGraph
END.
```

7

การแปลงภาพบนจอภาพ

เมื่อมีภาพปรากฏอยู่บนจอภาพ เราสามารถแปลงภาพนั้นให้มี ลักษณะ ขนาด ตำแหน่ง และจำนวน แตกต่างไปจากภาพเดิมได้ โดยอาจจะแปลงทั้งภาพหรือส่วนหนึ่งของภาพก็ได้ การแปลงภาพที่จะกล่าวถึงในบทนี้มี 7 ประการ ได้แก่ การแสดงซ้ำ การก๊อบปี้ การกลับสี การเลื่อน การเบ้ การกลับภาพ และการเปลี่ยนขนาดภาพ

วิธีการแปลงภาพทั้งหมดนี้สามารถใช้ได้กับภาพทุกภาพที่ปรากฏอยู่บนจอภาพ ไม่ว่าภาพนั้นจะเป็นแบบเวกเตอร์หรือแบบบิตแมพ เพื่อความสะดวกต่อการศึกษาในที่นี้จะใช้ภาพที่ได้จากไฟล์ BIRD.GTP และ PUPPY.GTP เพราะเป็นไฟล์ภาพที่เราสามารถสร้างขึ้นเองได้ตามวิธีการที่ได้กล่าวมาแล้วในบทที่ 3 สำหรับในบางตัวอย่างที่ใช้ไฟล์ LADY.GTP ถ้าไม่มีไฟล์นี้ก็สามารถใช้ไฟล์อื่นที่มีชนิดเป็น GTP แทนได้

7.1 การแสดงภาพซ้ำ

การแสดงภาพซ้ำ หมายถึง การนำภาพที่มีอยู่ในหน่วยความจำมาแสดงที่จอภาพซ้ำกันหลายครั้ง ซึ่งอาจจะแสดงที่ตำแหน่งเดิมหรือแสดงที่ตำแหน่งใหม่ก็ได้ โพรซีเยอร์แสดงภาพซ้ำได้แก่ **RepeatPicture** ซึ่งมีรูปแบบดังนี้

```
RepeatPicture (X, Y, NoOfPicture : Integer)
```

โดย X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

NoOfPicture คือจำนวนภาพที่แสดงซ้ำ

การใช้โพรซีเยอร์ RepeatPicture มีตัวอย่างดังนี้

ตัวอย่าง เราทราบแล้วว่า โปรแกรมแสดงภาพจากไฟล์ชนิด GTP (โปรแกรม 6.10) จะอ่านข้อมูลภาพจากไฟล์มาเก็บไว้ในหน่วยความจำ จากข้อมูลชุดนี้เราสามารถนำมาแสดงเป็นภาพ ณ ตำแหน่งต่างๆบนจอภาพได้ไม่จำกัดตำแหน่งและจำนวน โปรแกรม 7.1 ต่อไปนี้ ใช้สำหรับแสดงภาพจากไฟล์ชนิด GTP ได้ทุกไฟล์ โปรแกรมจะเริ่มด้วยการแสดงภาพแรกที่ตำแหน่งมุมบนซ้าย เมื่อกดคีย์ Enter โปรแกรมจะแสดงภาพเดิมซ้ำที่ตำแหน่งห่างไปทางขวามือ 100 พิกเซล โดยมีจำนวนภาพเท่ากับหรือน้อยกว่าจำนวนที่กำหนดไว้

โปรแกรม 7.1

```
PROGRAM PicGtpRepeat;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
   Assign(PictureFile,FileName);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;
```

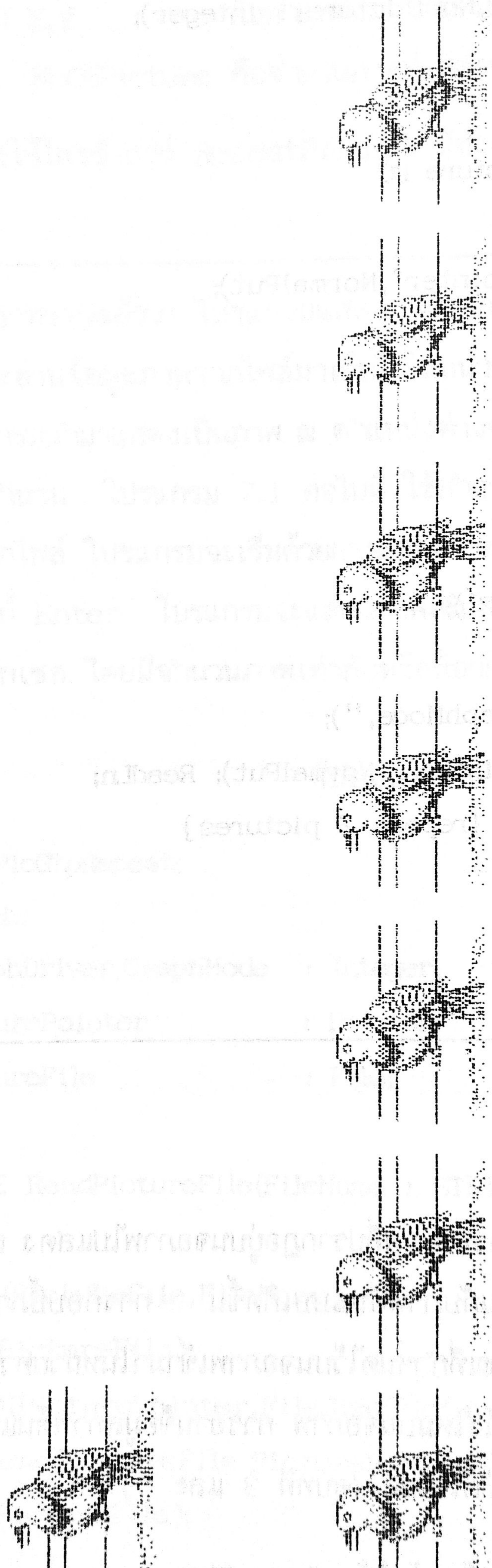
```
PROCEDURE RepeatPicture(X,Y,NoOfPicture : Integer);
VAR Count : Integer;
BEGIN
   FOR Count := 1 TO NoOfPicture DO
   BEGIN
      PutImage(X,Y,PicturePointer^,NormalPut);
      Inc(X,100);
   END;
END;

BEGIN
   ReadPictureFile('BIRD.GTP');
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0, 0, PicturePointer^, NormalPut); ReadLn;
   RepeatPicture(0, 100, 7); {repeat 7 pictures}
   ReadLn;
   CloseGraph
END.
```

7.2 การก๊อบปี้ภาพ

การก๊อบปี้ภาพ หมายถึง การนำภาพที่ปรากฏอยู่บนจอภาพไปแสดง ณ ตำแหน่งใหม่บนจอภาพ ทำให้มีภาพที่เหมือนกับภาพต้นแบบเกิดขึ้น การก๊อบปี้ภาพทำได้โดยการอ่านข้อมูลภาพต้นแบบจากบริเวณที่กำหนดไว้บนจอภาพเข้ามาในหน่วยความจำ แล้วนำข้อมูลภาพนั้นไปแสดง ณ ตำแหน่งใหม่บนจอภาพ การอ่านข้อมูลภาพต้นแบบมีวิธีการเหมือนกับการอ่านข้อมูลภาพเพื่อบันทึกในไฟล์ (ดูบทที่ 3 และ 6) คือ

1. กำหนดขอบเขตของภาพด้วยฟังก์ชัน ImageSize



รูปที่ 7.1 ภาพบนจอภาพที่ได้จากโปรแกรม 7.1

2. กำหนดขนาดของหน่วยความจำสำหรับเก็บภาพให้แก่แวเรียเบิล ซึ่งในตัวอย่างนี้คือ MemUsed

3. จองหน่วยความจำตามขนาดที่กำหนดด้วยโพรซีเยอร์ GetMem

4. อ่านภาพจากจอภาพเข้ามาในหน่วยความจำด้วยโพรซีเยอร์ GetImage

5. เมื่อมีภาพอยู่ในหน่วยความจำแล้วจึงแสดงภาพนั้น ณ ตำแหน่งใหม่ ด้วยโพรซีเยอร์ PutImage

โพรซีเยอร์ก๊อบปี้ภาพได้แก่ **CopyPicture** ซึ่งมีรูปแบบดังนี้

CopyPicture (X1, Y1, X2, Y2, X, Y : Integer)

โดย X1,Y1 คือตำแหน่งมุมซ้ายบนของภาพต้นแบบ
X2,Y2 คือตำแหน่งมุมขวาล่างของภาพต้นแบบ
X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

ตัวอย่างการก๊อบปี้ภาพมีดังนี้

ตัวอย่าง โปรแกรม 7.2 แสดงการก๊อบปี้ภาพจาก 1 ภาพให้เป็น 5 ภาพ โปรแกรมเริ่มด้วยการแสดงภาพจากไฟล์ PUPPY.GTP 1 ภาพและแสดงภาพจากไฟล์ BIRD.GTP อีก 1 ภาพ ต่อจากนั้นเราจึงก๊อบปี้ภาพแรก โดยการอ่านข้อมูลจากภาพบนจอภาพ ซึ่งเป็นภาพสุนัขเข้ามาในหน่วยความจำของคอมพิวเตอร์ แล้วจึงนำข้อมูลที่มีอยู่ไปแสดงบนจอภาพ ณ ตำแหน่งต่างๆ ตามที่กำหนดไว้

โปรแกรม 7.2

```
PROGRAM CopyPicFromScreen;
USES Graph;
VAR  GraphDriver,GraphMode  : Integer;
```

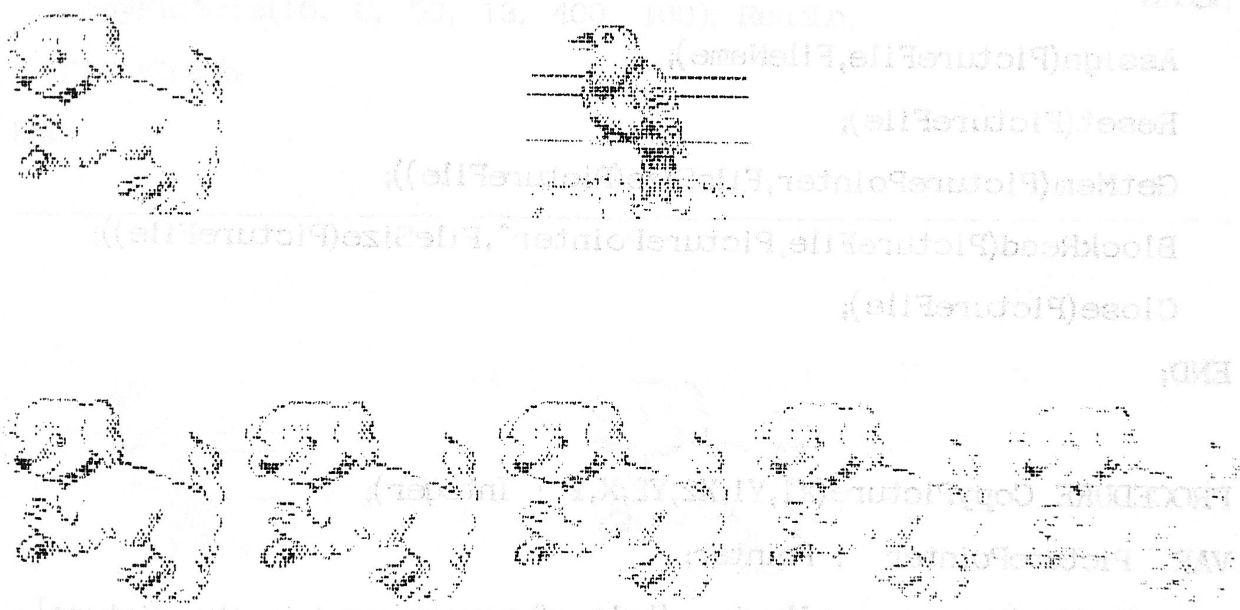
```
   PicturePointer          : Pointer;
   PictureFile             : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
  Assign(PictureFile,FileName);
  Reset(PictureFile);
  GetMem(PicturePointer,FileSize(PictureFile));
  BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
  Close(PictureFile);
END;

PROCEDURE CopyPicture(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;    {byte of memory used in the picture}
BEGIN
  MemUsed := ImageSize(X1,Y1,X2,Y2);
  GetMem(PicturePointer,MemUsed);
  GetImage(X1,Y1,X2,Y2,PicturePointer^);
  PutImage(X,Y,PicturePointer^,NormalPut);
END;

BEGIN
  ReadPictureFile('PUPPY.GTP');
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  ClearDevice;
  PutImage(0,0,PicturePointer^,NormalPut); Readln;
  ReadPictureFile('BIRD.GTP');
  PutImage(200,0,PicturePointer^,NormalPut); Readln;
```

```
    CopyPicture(0, 0, 87, 51, 000, 100);
    CopyPicture(0, 0, 87, 51, 100, 100);
    CopyPicture(0, 0, 87, 51, 200, 100);
    CopyPicture(0, 0, 87, 51, 300, 100);
    CopyPicture(0, 0, 87, 51, 400, 100); ReadLn;
    CloseGraph
END.
```



รูปที่ 7.2 ภาพบนจอภาพที่ได้จากโปรแกรม 7.2

ตัวอย่าง โปรแกรม 7.3 แสดงการก๊อบปี้ภาพเต็มภาพและจากบางส่วนของภาพต้นแบบ ในที่นี้จะใช้ภาพนกจากไฟล์ BIRD.GTP มาเป็นตัวอย่าง โดยในบางภาพเราจะก๊อบปี้เฉพาะภาพส่วนหัวของนกมาแสดง แต่บางภาพจะก๊อบปี้มาแสดงเต็มภาพ โปรแกรมเริ่มด้วยการแสดงภาพนกเต็มตัว 1 ภาพ เมื่อกดคีย์ Enter จะมีการก๊อบปี้ภาพบางส่วน และ ภาพเต็มภาพมาแสดงที่ด้านล่างของภาพเดิม รวม 5 ภาพ

โปรแกรม 7.3

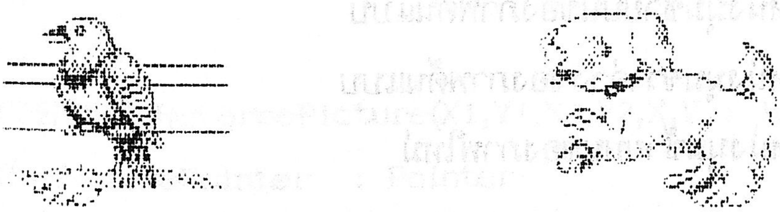
```
PROGRAM CopyPartOfPicFromScreen;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;

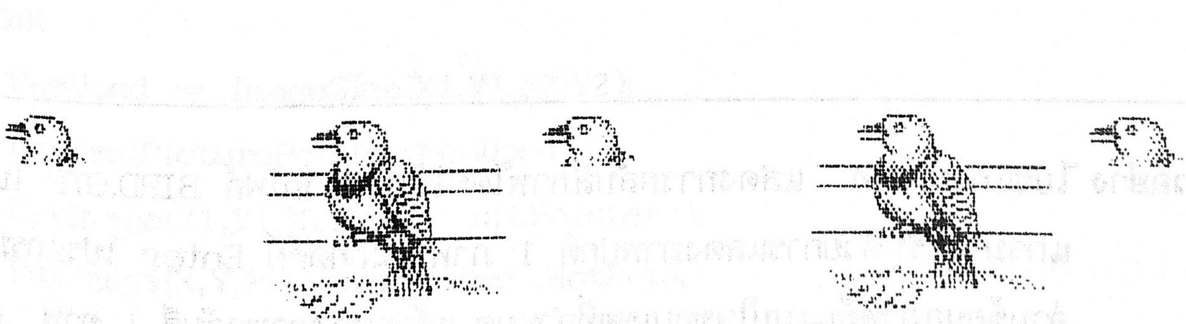
PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
   Assign(PictureFile,FileName);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;


PROCEDURE CopyPicture(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;    {byte of memory used in the picture}
BEGIN
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   GetImage(X1,Y1,X2,Y2,PicturePointer^);
   PutImage(X,Y,PicturePointer^,NormalPut);
END;


BEGIN
   ReadPictureFile('BIRD.GTP');
   GraphDriver := Detect;
```

```
  InitGraph(GraphDriver,GraphMode,'');
  ClearDevice;
  PutImage(0,0,PicturePointer^,NormalPut); Readln;
  ReadPictureFile('PUPPY.GTP');
  PutImage(200,0,PicturePointer^,NormalPut); Readln;
  CopyPicture(16, 0, 50, 13, 000, 100);
  CopyPicture( 0, 0, 87, 51, 100, 100);
  CopyPicture(16, 0, 50, 13, 200, 100);
  CopyPicture( 0, 0, 87, 51, 300, 100);
  CopyPicture(16, 0, 50, 13, 400, 100); ReadLn;
  CloseGraph
END.
```





รูปที่ 7.3 ภาพบนจอภาพที่ได้จากโปรแกรม 7.3

7.3 การกลับสีภาพ

การกลับสีภาพ หมายถึง การเปลี่ยนสีภาพให้เป็นสีพื้นและเปลี่ยนสีพื้นให้เป็นสีภาพ กรณีจอภาพสีเดียว การกลับสีภาพจะมีผลให้จุดสว่างเปลี่ยนเป็นไม่มีจุดและบริเวณที่ไม่มีจุดจะเป็นจุดสว่าง การกลับสีจะมีผลต่อภาพเฉพาะในขอบเขตที่กำหนดไว้เท่านั้น เพราะฉะนั้นการกลับสีอาจจะทำทั้งภาพหรือส่วนหนึ่งของภาพก็ได้

การกลับสีภาพสามารถได้ง่ายๆ โดยกำหนดพารามิเตอร์ลักษณะการแสดงภาพของโพรซีเยอร์ PutImage ซึ่งเป็นพารามิเตอร์ตัวที่ 4 ให้เป็น NotPut หรือ 4 ดังต่อไปนี้

```
PutImage (X, Y, PixelPointer, NotPut)
```

โพรซีเยอร์กลับสีภาพได้แก่ **ReversePicture** ซึ่งมีรูปแบบดังนี้

```
ReversePicture (X1, Y1, X2, Y2, X, Y : Integer)
```

โดย X1,Y1 คือตำแหน่งมุมซ้ายบนของภาพต้นแบบ

X2,Y2 คือตำแหน่งมุมขวาล่างของภาพต้นแบบ

X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

ตัวอย่างการกลับสีภาพมีดังนี้

ตัวอย่าง โปรแกรม 7.4 แสดงการกลับสีภาพโดยใช้ภาพจากไฟล์ BIRD.GTP โปรแกรมจะเริ่มด้วยการแสดงภาพปกติ 1 ภาพ เมื่อกดคีย์ Enter โปรแกรมจะอ่านข้อมูลภาพต้นแบบในขอบเขตที่กำหนด แล้วแสดงภาพกลับสี 1 ภาพ เมื่อกดคีย์ Enter อีกครั้ง โปรแกรมจะอ่านข้อมูลภาพจากภาพก่อน แล้วแสดงภาพกลับสีอีก 1 ภาพ สลับกันเช่นนี้จนครบ 6 ภาพ เพื่อเปรียบเทียบลักษณะของภาพต้นแบบกับภาพกลับสี

โปรแกรม 7.4

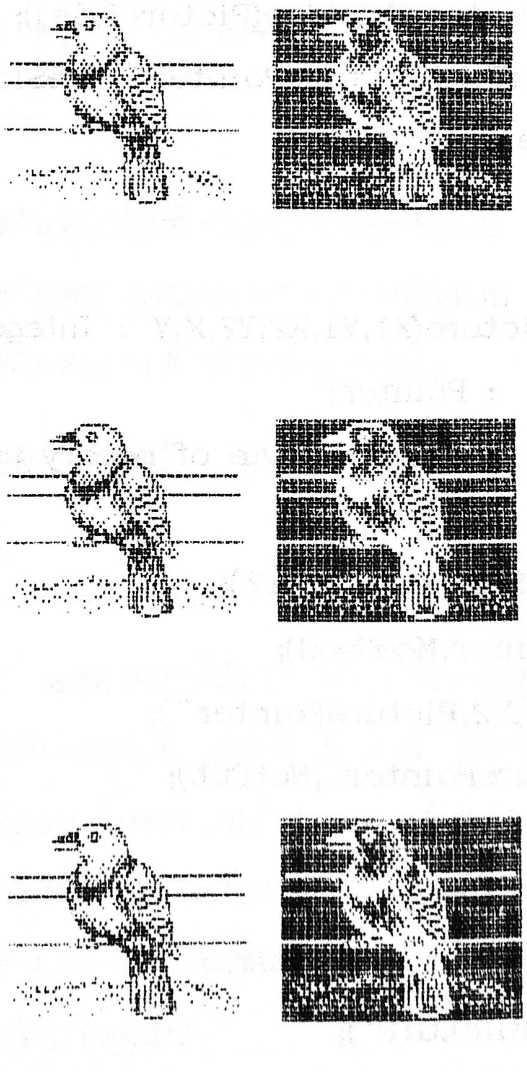
```
PROGRAM ReversePictureOnScreen;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
   Assign(PictureFile,FileName);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;

PROCEDURE ReversePicture(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;    {byte of memory used in the picture}
BEGIN
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   GetImage(X1,Y1,X2,Y2,PicturePointer^);
   PutImage(X,Y,PicturePointer^,NotPut);
END;

BEGIN
   ReadPictureFile('BIRD.GTP');
   GraphDriver := Detect;
```

```
  InitGraph(GraphDriver,GraphMode,'');
  PutImage(0, 0, PicturePointer^, NormalPut); ReadLn;
  ReversePicture(  0, 0,  87, 51, 100, 0); ReadLn;
  ReversePicture(100, 0, 187, 51, 200, 0); ReadLn;
  ReversePicture(200, 0, 287, 51, 300, 0); ReadLn;
  ReversePicture(300, 0, 387, 51, 400, 0); ReadLn;
  ReversePicture(400, 0, 487, 51, 500, 0); ReadLn;
  CloseGraph
END.
```



รูปที่ 7.4 ภาพบนจอภาพที่ได้จากโปรแกรม 7.4

7.4 การเลื่อนภาพ

การเลื่อนภาพ หมายถึง การนำภาพซึ่งเคยอยู่ ณ ที่หนึ่งบนจอภาพ ไปอยู่ที่ตำแหน่งอื่น ความแตกต่างของการเลื่อนภาพและการก๊อบปี้ภาพก็คือ การก๊อบปี้ทำให้เกิดภาพเพิ่มขึ้นโดยที่ภาพต้นแบบก็ยังคงอยู่ ณ ที่เดิม แต่การเลื่อนภาพจะไม่ทำให้ภาพเพิ่มขึ้น เป็นเพียงการนำภาพเดิมไปแสดง ณ ตำแหน่งใหม่ และภาพซึ่งเคยอยู่ ณ ที่เดิมจะหายไปจากบริเวณนั้น

การเลื่อนภาพมีวิธีการคล้ายกับการก๊อบปี้ภาพ แต่เพิ่มการลบภาพที่ตำแหน่งเดิมขึ้นอีก 1 ขั้นตอน วิธีลบภาพที่ตำแหน่งเดิมทำได้โดยการใช้สีพื้น (background color) ของจอภาพระบายที่บริเวณนั้น ซึ่งจะมีผลให้ภาพเดิมหายไป โพรซีเยอร์ที่ใช้คือ **Bar** ซึ่งมีรูปแบบดังนี้

```
Bar (X1, Y1, X2, Y2 : Integer)
```

โดย X1,Y1 เป็นตำแหน่งมุมบนซ้ายของภาพสี่เหลี่ยม

X2,Y2 เป็นตำแหน่งมุมล่างขวาของภาพสี่เหลี่ยม

Bar จะระบายสีจนเต็มบริเวณสี่เหลี่ยมที่กำหนดด้วย X1,Y1 และ X2,Y2 ด้วยสีที่เคยกำหนดไว้ เพื่อให้ Bar ใช้สีตรงกับความต้องการ จึงต้องกำหนดสีไว้ล่วงหน้าด้วยโพรซีเยอร์ **SetFillStyle** ซึ่งมีรูปแบบดังนี้

```
SetFillStyle (Pattern : Word; Color : Word)
```

โดย Pattern คือ ลักษณะลายที่กำหนดให้แก่ Bar ในกรณีนี้กำหนดให้เป็นลายว่างคือ EmptyFill หรือ 0 ส่วน Color คือสีที่กำหนดให้แก่ Bar ในกรณีนี้กำหนดเป็นสีพื้น ซึ่งทำได้โดยใช้ฟังก์ชัน GetBkColor ดังนั้นพารามิเตอร์ของโพรซีเยอร์ SetFillStyle จึงเป็นดังนี้

```
SetFillStyle (EmptyFill,GetBkColor)
```

โพรซีเยอร์เลื่อนภาพได้แก่ **MovePicture** ซึ่งมีรูปแบบดังนี้

```
MovePicture (X1, Y1, X2, Y2, X, Y : Integer)
```

โดย X1,Y1 คือตำแหน่งมุมซ้ายบนของภาพ

X2,Y2 คือตำแหน่งมุมขวาล่างของภาพ

X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

ตัวอย่างการเลื่อนภาพมีดังนี้

ตัวอย่าง โปรแกรม 7.5 แสดงการเลื่อนภาพไปที่ตำแหน่งใหม่บนจอภาพโดยใช้ภาพจากไฟล์ PUPPY.GTP โปรแกรมเริ่มด้วยการอ่านข้อมูลภาพจากไฟล์ แสดงภาพที่จอภาพและรอการกดคีย์ เมื่อกดคีย์ Enter ภาพจะเลื่อนไปที่ตำแหน่งใหม่ตามที่กำหนดไว้

โปรแกรม 7.5

```
PROGRAM MovePictureOnScreen;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
   Assign(PictureFile,FileName);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;

PROCEDURE MovePicture(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer  : Pointer;
```

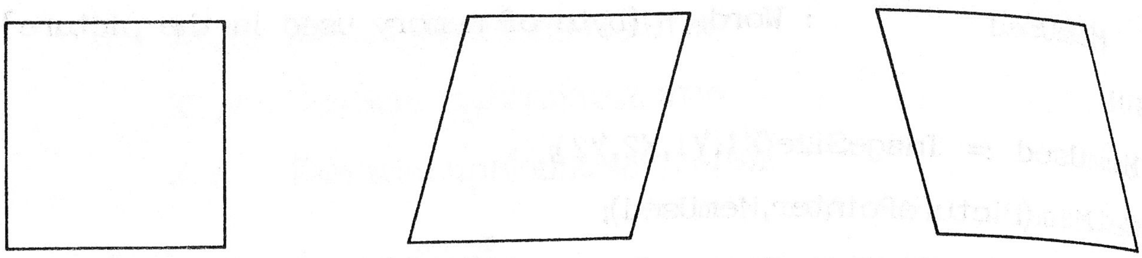
```
    MemUsed          : Word;    {byte of memory used in the picture}
BEGIN
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   GetImage(X1,Y1,X2,Y2,PicturePointer^);
   PutImage(X,Y,PicturePointer^,NormalPut);
   SetFillStyle(EmptyFill,GetBkColor);
   Bar(X1,Y1,X2,Y2);
END;

BEGIN
   ReadPictureFile('PUPPY.GTP');
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut); Readln;
  MovePicture(  0,    0,  87,   51, 500,    0); ReadLn;
  MovePicture(500,    0, 587,   51, 100,  100); ReadLn;
  MovePicture(100,  100, 187,  151, 200,    0); ReadLn;
  CloseGraph
END.
```

7.5 การเบ้ภาพ

การเบ้ภาพ หมายถึง การทำให้ภาพแตกต่างจากภาพเดิมในลักษณะเบ้ การเบ้ภาพสามารถทำได้ 2 แนวคือ การเบ้ภาพในแนวนอนและการเบ้ภาพในแนวตั้ง ซึ่งมีรายละเอียดดังต่อไปนี้

7.5.1 การเบ้ภาพในแนวนอน หรือการเบ้ภาพในแนวแกน X หมายถึง การทำให้ภาพเบ้ไปทางขวามือหรือทางซ้ายมือดังแสดงในรูปที่ 7.5



รูปที่ 7.5 การเบ้ภาพในแนวนอน

การเบ้ภาพในแนวนอนทำได้ โดยการจัดแต่ละแถวที่มีอยู่ในภาพให้เรียงเหลื่อมกันตามลำดับ ซึ่งสามารถจัดได้ 2 แบบคือ ให้เหลื่อมไปทางขวาสำหรับการเบ้ภาพทางขวา และให้เหลื่อมไปทางซ้ายสำหรับการเบ้ภาพทางซ้าย ซึ่งมีรายละเอียดดังต่อไปนี้

1. การเบ้ภาพทางขวา เราจัดให้แถวล่างเริ่มต้น ณ ตำแหน่งที่ห่างไปทางขวาของจุดเริ่มต้นของแถวบน เรียงกันตามลำดับจนครบทุกแถว ระยะเหลื่อมของแถวบนกับแถวล่างจะมีผลให้ภาพเบ้ไม่เท่ากัน นั่นคือถ้าระยะเหลื่อมน้อยภาพจะเบ้น้อย และถ้าระยะเหลื่อมมากภาพจะเบ้มาก

2. การเบ้ภาพทางซ้าย เราจะแสดงเส้นในทิศทางตรงกันข้ามกับการเบ้ภาพทางขวาคือ จัดให้แถวล่างเริ่มต้นที่ตำแหน่งห่างไปทางซ้ายของจุดเริ่มต้นของแถวบนและเรียงกันตามลำดับจนครบทุกเส้นของภาพนั้น

ไพรซีเยอร์เบ้ภาพในแนวนอนคือ **ShearPictureX** ซึ่งมีรูปแบบดังนี้

```
ShearPictureX (X1, Y1, X2, Y2, X, Y : Integer;
               Direction : Char)
```

โดย X1,Y1 คือตำแหน่งมุมซ้ายบนของภาพต้นแบบ

X2,Y2 คือตำแหน่งมุมขวาล่างของภาพต้นแบบ

X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

Direction คือทิศทางการเบ้ภาพซึ่งกำหนดไว้ดังนี้

R หรือ r เบ้ภาพไปทางขวา

L หรือ l เบ้ภาพไปทางซ้าย

ตัวอย่างการเบ้ภาพในแนวนอนมีดังนี้

ตัวอย่าง โปรแกรม 7.6 แสดงการเบ้ภาพในแนวนอนโดยใช้ภาพจากไฟล์ BIRD.GTP โปรแกรมจะเริ่มด้วยการแสดงภาพปกติ เมื่อกดคีย์ Enter จะแสดงภาพเบ้ไปทางขวา และเมื่อกดคีย์ Enter อีก จะแสดงภาพเบ้ไปทางซ้าย

โปรแกรม 7.6

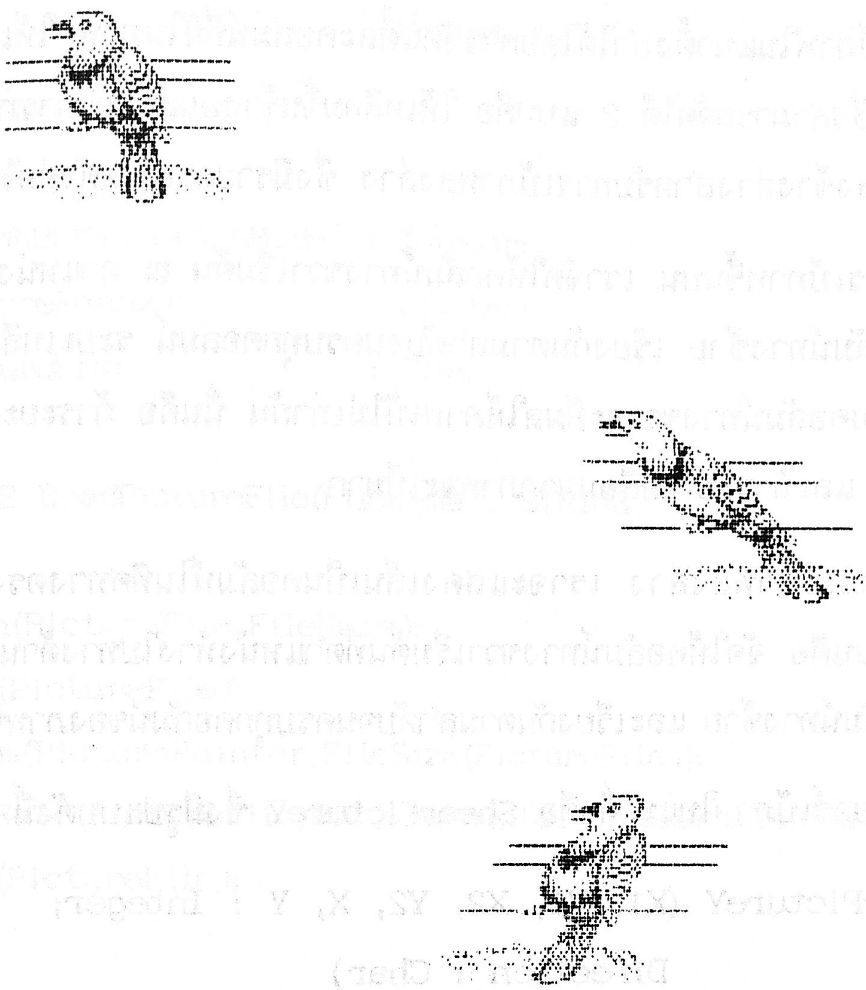
```
PROGRAM ShearXPicture;
USES Graph;
VAR  GraphDriver,GraphMode    : Integer;
     PicturePointer           : Pointer;
     PictureFile              : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
  Assign(PictureFile,FileName);
  Reset(PictureFile);
  GetMem(PicturePointer,FileSize(PictureFile));
  BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
  Close(PictureFile);
END;

PROCEDURE ShearPictureX(X1,Y1,X2,Y2,X,Y : Integer; Direction : Char);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;    {byte of memory used in the picture}
     Row : Integer;
```

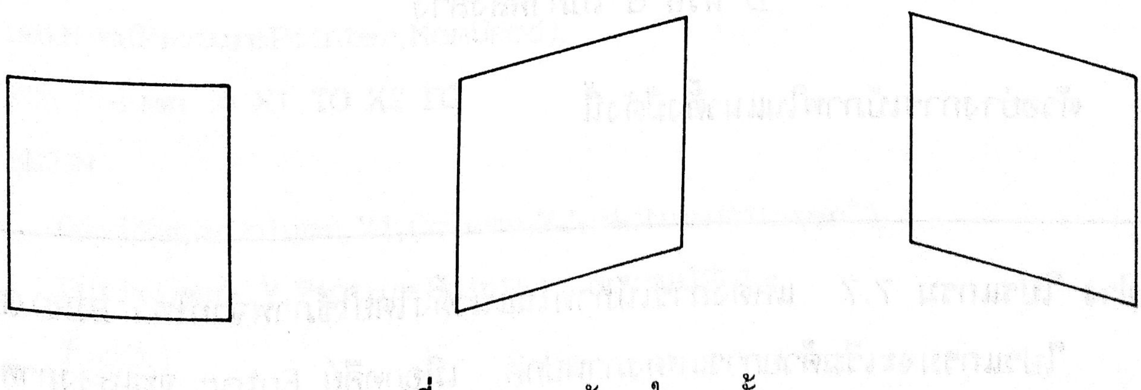
```
BEGIN
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   FOR Row := Y1 TO Y2 DO
   BEGIN
      GetImage(X1,Row,X2,Row,PicturePointer^);
      PutImage(X,Y,PicturePointer^,NormalPut);
      CASE Direction OF
         'R','r'  :  Inc(X); {shear right}
         'L','l'  :  Dec(X)  {shear left}
      END;
      Inc(Y);
   END;
END;

BEGIN
   ReadPictureFile('BIRD.GTP');
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut); Readln;
   ShearPictureX(0, 0, 87, 51, 200,100,'r'); ReadLn;
   ShearPictureX(0, 0, 87, 51, 200,200,'l'); Readln;
   CloseGraph
END.
```



รูปที่ 7.6 ภาพบนจอภาพที่ได้จากโปรแกรม 7.6

7.5.2 การเบ้ภาพในแนวตั้ง หรือการเบ้ภาพในแนวแกน Y หมายถึง การทำให้ภาพเบ้ขึ้นข้างบนหรือเบ้ลงข้างล่าง ดังแสดงในรูปที่ 7.7



รูปที่ 7.7 การเบ้ภาพในแนวตั้ง

การเบ้ภาพในแนวตั้งทำได้โดยการจัดแต่ละคอลัมน์ที่มีในภาพ ให้เรียงเหลื่อมกันตามลำดับ ซึ่งสามารถจัดได้ 2 แบบคือ ให้เหลื่อมขึ้นข้างบนสำหรับการเบ้ภาพขึ้นบน และให้เหลื่อมลงข้างล่างสำหรับการเบ้ภาพลงล่าง ซึ่งมีรายละเอียดดังต่อไปนี้

1. การเบ้ภาพขึ้นบน เราจัดให้คอลัมน์ทางขวาเริ่มต้น ณ ตำแหน่งที่สูงกว่าจุดเริ่มต้นของคอลัมน์ทางซ้าย เรียงกันตามลำดับจนครบทุกคอลัมน์ ระยะเหลื่อมของคอลัมน์ทางซ้ายกับคอลัมน์ทางขวาจะมีผลให้ภาพเบ้ไม่เท่ากัน นั่นคือ ถ้าระยะเหลื่อมน้อยภาพจะเบ้น้อย และถ้าระยะเหลื่อมมากภาพจะเบ้มาก

2. การเบ้ภาพลงล่าง เราจะแสดงเส้นเป็นคอลัมน์ในทิศทางตรงกันข้ามกับการเบ้ภาพขึ้นบนคือ จัดให้คอลัมน์ทางขวาเริ่มต้นที่ตำแหน่งห่างไปทางด้านล่างของจุดเริ่มต้นของคอลัมน์ทางซ้าย และเรียงกันตามลำดับจนครบทุกคอลัมน์ของภาพนั้น

ไพรซีเยอร์เบ้ภาพในแนวตั้งคือ **ShearPictureY** ซึ่งมีรูปแบบดังนี้

```
ShearPictureY (X1, Y1, X2, Y2, X, Y : Integer;
                   Direction : Char)
```

โดย X1,Y1 คือตำแหน่งมุมซ้ายบนของภาพต้นแบบ

X2,Y2 คือตำแหน่งมุมขวาล่างของภาพต้นแบบ

X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

Direction คือทิศทางการเบ้ภาพซึ่งกำหนดไว้ดังนี้

U หรือ u เบ้ภาพขึ้นบน

D หรือ d เบ้ภาพลงล่าง

ตัวอย่างการเบ้ภาพในแนวตั้งมีดังนี้

ตัวอย่าง โปรแกรม 7.7 แสดงการเบ้ภาพในแนวตั้งโดยใช้ภาพจากไฟล์ BIRD.GTP โปรแกรมจะเริ่มด้วยการแสดงภาพปกติ เมื่อกดคีย์ Enter จะแสดงภาพเบ้ขึ้นบน และเมื่อกดคีย์ Enter อีก จะแสดงภาพเบ้ลงล่าง

โปรแกรม 7.7

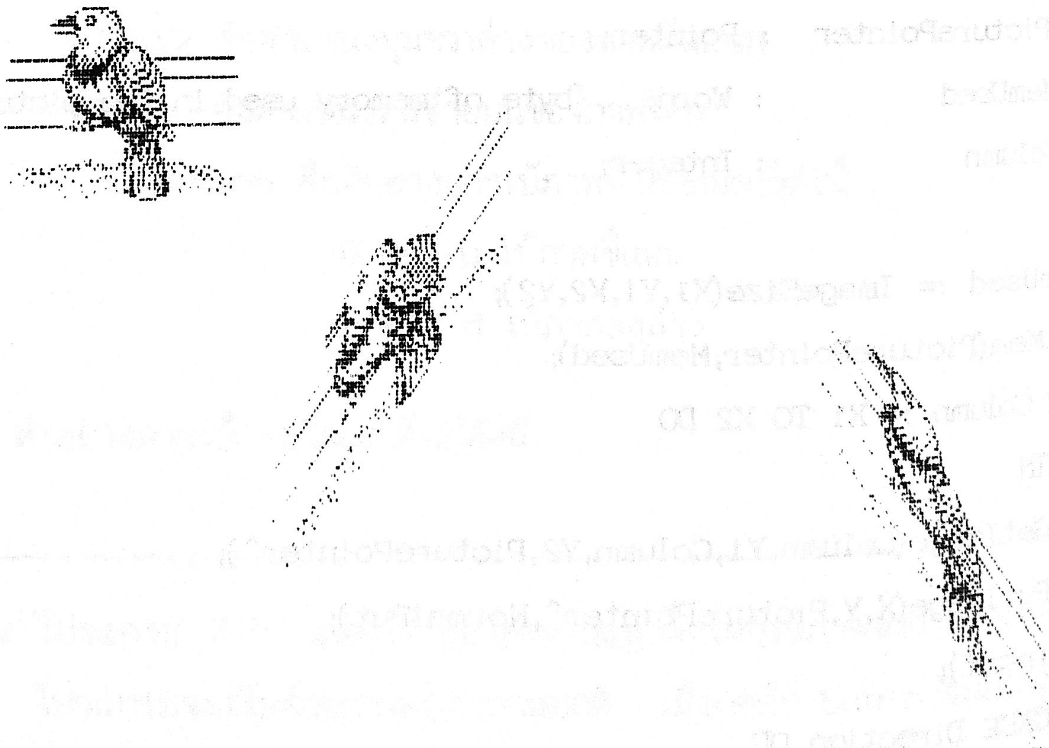
```
PROGRAM ShearYPicture;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
   Assign(PictureFile,FileName);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;

PROCEDURE ShearPictureY(X1,Y1,X2,Y2,X,Y : Integer; Direction : Char);
VAR  PicturePointer    : Pointer;
     MemUsed           : Word;    {byte of memory used in the picture}
     Column            : Integer;
BEGIN
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   FOR Column := X1 TO X2 DO
   BEGIN
      GetImage(Column,Y1,Column,Y2,PicturePointer^);
      PutImage(X,Y,PicturePointer^,NormalPut);
      Inc(X);
      CASE Direction OF
```

```
        'U','u'  :  Dec(Y); {shear up}
        'D','d'  :  Inc(Y)  {shear down}
      END
   END;
END;

BEGIN
   ReadPictureFile('BIRD.GTP');
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut); Readln;
   ShearPictureY(0,0,87,51,100,100,'u'); ReadLn;
   ShearPictureY(0,0,87,51,300,100,'d'); Readln;
   CloseGraph
END.
```

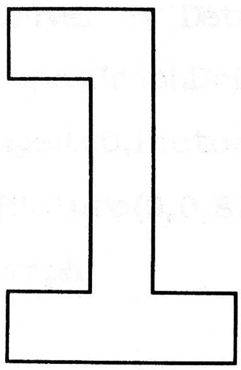


รูปที่ 7.8 ภาพบนจอภาพที่ได้จากโปรแกรม 7.7

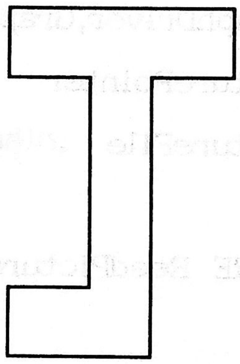
7.6 การกลับภาพ

การกลับภาพ หมายถึง การทำให้ทิศทางของภาพใหม่กลับกับทิศทางของภาพเดิม การกลับภาพสามารถทำได้ 2 ลักษณะคือ การกลับภาพแบบบน-ล่าง และการกลับภาพแบบซ้าย-ขวา ซึ่งมีรายละเอียดดังต่อไปนี้

7.6.1 การกลับภาพแบบบน-ล่าง หรือการกลับหัวภาพ หมายถึงการกลับเอาด้านล่างของภาพขึ้นอยู่ข้างบน และกลับเอาด้านบนของภาพลงอยู่ข้างล่าง ดังแสดงในรูปที่ 7.9



ภาพปกติ



ภาพกลับบน-ล่าง

รูปที่ 7.9 การกลับภาพแบบบน-ล่าง

การกลับภาพแบบบน-ล่าง ทำได้โดยการอ่านข้อมูลภาพจากภาพเดิมทีละแถวเรียงลำดับจากด้านล่างขึ้นข้างบน แล้วนำมาแสดงเป็นภาพใหม่ โดยแสดงทีละแถวเรียงลำดับจากด้านบนลงข้างล่าง ด้วยวิธีนี้ จะทำให้ด้านบนของภาพเดิมกลายเป็นด้านล่างของภาพใหม่ ซึ่งเราจะเห็นภาพมีลักษณะหัวกลับกับภาพเดิม

โพรซีเยอร์กลับภาพแบบบน-ล่าง คือ **UpEndPicture** ซึ่งมีรูปแบบดังนี้

```
UpEndPicture (X1, Y1, X2, Y2, X, Y : Integer)
```

โดย X1,Y1 คือตำแหน่งมุมซ้ายบนของภาพต้นแบบ

X2,Y2 คือตำแหน่งมุมขวาล่างของภาพต้นแบบ

X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

ตัวอย่างการกลับภาพแบบบน-ล่างมีดังนี้

ตัวอย่าง โปรแกรม 7.8 แสดงภาพกลับบน-ล่าง ซึ่งเป็นภาพจากไฟล์ PUPPY.GTP โปรแกรมเริ่มด้วยการแสดงภาพปกติ เมื่อกดคีย์ Enter โปรแกรมจะแสดงภาพกลับบน-ล่าง

โปรแกรม 7.8

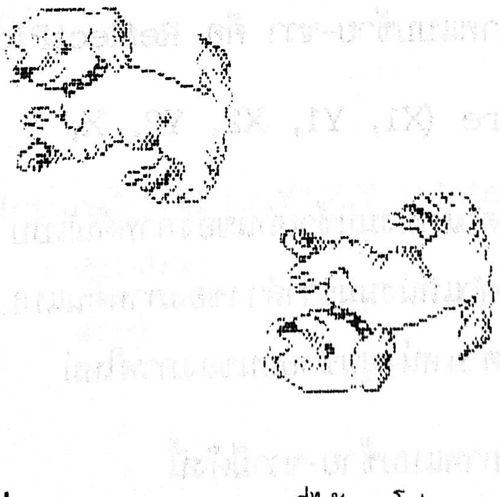
```
PROGRAM UpEndPictureFromGtp;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
   Assign(PictureFile,FileName);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;

PROCEDURE UpEndPicture(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;
     Row : Integer;
BEGIN
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
```

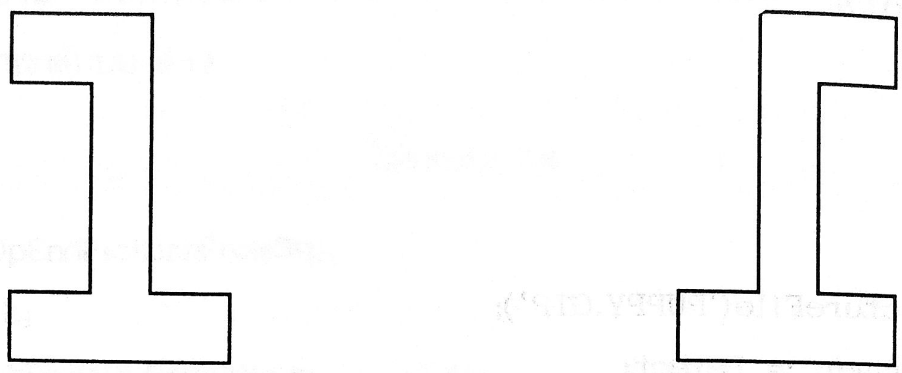
```
   FOR Row := Y2 DOWNTO Y1 DO
   BEGIN
      GetImage(X1,Row,X2,Row,PicturePointer^);
      PutImage(X,Y,PicturePointer^,NormalPut);
      Inc(Y);
   END;
END;

BEGIN
   ReadPictureFile('PUPPY.GTP');
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut); Readln;
   UpEndPicture(0,0,87,51,200,150);    Readln;
   CloseGraph
END.
```



รูปที่ 7.10 ภาพบนจอภาพที่ได้จากโปรแกรม 7.8

7.6.2 การกลับภาพแบบซ้าย-ขวา หรือการสะท้อนภาพ หมายถึง การกลับเอาด้านซ้ายของภาพไปเป็นด้านขวา และกลับเอาด้านขวามาเป็นด้านซ้าย ดังแสดงในรูปที่ 7.11



รูปที่ 7.11 การกลับภาพแบบซ้าย-ขวา

การกลับภาพแบบซ้าย-ขวา ทำได้โดยการอ่านข้อมูลภาพจากภาพเดิมทีละคอลัมน์เรียงลำดับจากด้านขวามาด้านซ้าย แล้วนำมาแสดงเป็นภาพใหม่ โดยแสดงทีละคอลัมน์เรียงลำดับจากด้านซ้ายไปด้านขวา ด้วยวิธีนี้จะทำให้ด้านขวาของภาพเดิมกลายเป็นด้านซ้ายของภาพใหม่ ซึ่งเราจะเห็นภาพมีลักษณะซ้าย-ขวากลับกับภาพเดิม

โพรซีเยอร์กลับภาพแบบซ้าย-ขวา คือ **ReflectPicture** ซึ่งมีรูปแบบดังนี้

```
ReflectPicture (X1, Y1, X2, Y2, X, Y : Integer)
```

โดย X1,Y1 คือตำแหน่งมุมซ้ายบนของภาพต้นแบบ
X2,Y2 คือตำแหน่งมุมขวาล่างของภาพต้นแบบ
X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

ตัวอย่างการกลับภาพแบบซ้าย-ขวามีดังนี้

ตัวอย่าง โปรแกรม 7.9 แสดงภาพกลับซ้าย–ขวา ซึ่งเป็นภาพจากไฟล์ PUPPY.GTP โปรแกรมเริ่มด้วยการแสดงภาพปกติ เมื่อกดคีย์ Enter โปรแกรมจะแสดงภาพกลับซ้าย–ขวา

โปรแกรม 7.9

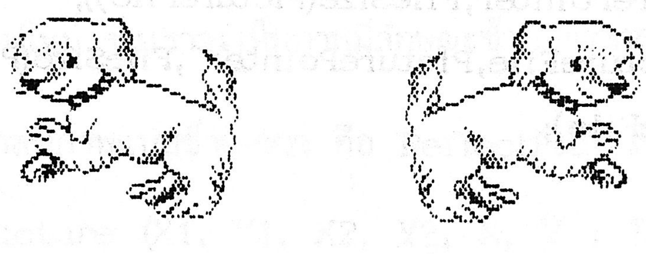
```
PROGRAM ReflectPictureFromGtp;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
   Assign(PictureFile,FileName);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;

PROCEDURE ReflectPicture(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;
     Col : Integer;
BEGIN
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   FOR Col := X2 DOWNTO X1 DO
```

```
  BEGIN
     GetImage(Col,Y1,Col,Y2,PicturePointer^);
     PutImage(X,Y,PicturePointer^,NormalPut);
     Inc(X);
  END;
END;

BEGIN
  ReadPictureFile('PUPPY.GTP');
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  PutImage(0,0,PicturePointer^,NormalPut); Readln;
  ReflectPicture(0,0,87,51,200,150);    Readln;
  CloseGraph
END.
```



รูปที่ 7.12 ภาพบนจอภาพที่ได้จากโปรแกรม 7.9

ตัวอย่าง การกลับภาพแบบซ้าย-ขวา บางครั้งอาจจะทำให้เกิดภาพแปลกๆขึ้นได้ ดังตัวอย่างการแสดงภาพจากไฟล์ LADY.GTP ในโปรแกรม 7.10 นี้ ซึ่งเริ่มด้วยการแสดงภาพธรรมดา เมื่อกดคีย์ Enter โปรแกรมจะแสดงภาพกลับซ้าย-ขวาที่มีลักษณะแปลกไปกว่าภาพเดิมมาก ทั้งนี้เป็นเพราะการกำหนดค่าให้แก่โพรซีเยอร์กลับภาพ ถ้าค่าเหล่านั้นเปลี่ยน ภาพก็จะเปลี่ยนไปด้วย

โปรแกรม 7.10

```
PROGRAM ReflectLadyGtp;
USES Graph;
VAR  GraphDriver,GraphMode    : Integer;
     PicturePointer           : Pointer;
     PictureFile              : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
   Assign(PictureFile,FileName);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;

PROCEDURE ReflectPicture(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;    {byte of memory used in the picture}
     Col : Integer;
BEGIN
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   FOR Col := X2 DOWNTO X1 DO
   BEGIN
      GetImage(Col,Y1,Col,Y2,PicturePointer^);
      PutImage(X,Y,PicturePointer^,NormalPut);
      Inc(X);
   END;
END;
```

```
BEGIN
   ReadPictureFile('LADY.GTP');
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut); Readln;
   ReflectPicture(8,0,519,284,200,0);   Readln;
   CloseGraph
END.
```

7.7 การเปลี่ยนขนาดภาพ

การเปลี่ยนขนาดภาพ หมายถึง การทำให้ขนาดของภาพแตกต่างไปจากเดิมซึ่งสามารถทำได้ 2 แบบคือ การขยายภาพและการย่อภาพ ซึ่งมีรายละเอียดดังต่อไปนี้

7.7.1 การขยายภาพ หมายถึง การทำให้ภาพใหม่มีขนาดโตกว่าภาพเดิม การขยายภาพทำได้ 3 แบบคือ การขยายภาพในทางสูง การขยายภาพในทางกว้าง และการขยายภาพทั้งในทางสูงและทางกว้าง ซึ่งมีรายละเอียดดังนี้

1. การขยายภาพในทางสูง หมายถึง การทำให้ภาพใหม่มีความสูงมากกว่าภาพเดิมแต่ความกว้างยังคงเท่าภาพเดิม

2. การขยายภาพในทางกว้าง หมายถึง การทำให้ภาพใหม่มีความกว้างมากกว่าภาพเดิมแต่ความสูงยังคงเท่าภาพเดิม

3. การขยายภาพทั้งในทางสูงและทางกว้าง หมายถึง การทำให้ภาพใหม่มีความกว้างและความสูงมากกว่าภาพเดิม การขยายภาพในลักษณะนี้ สามารถจัดสัดส่วนให้แตกต่างกันได้อีก 3 แบบคือ ขยายในทางสูงและทางกว้างในสัดส่วนเท่ากัน ขยายในทางสูงมากกว่าทางกว้าง และขยายในทางสูงน้อยกว่าทางกว้าง ซึ่งมีผลให้ภาพที่ได้จากการขยายมีลักษณะแตกต่างกัน



รูปที่ 7.13 ภาพบนจอภาพที่ได้จากโปรแกรม 7.10

การขยายภาพทุกแบบมีหลักการเหมือนกันคือ ตรวจหาจุดในแต่ละแถว ถ้าพบจุดก็จัดการขยายให้ได้ตามสัดส่วนที่ต้องการ แล้วแสดงจุดนั้น ณ ตำแหน่งที่สอดคล้องกับภาพเดิม การขยายจุดทำได้ด้วยโพรซีเยอร์ Bar ซึ่งมีรูปแบบดังนี้

```
Bar (X1, Y1, X2, Y2 : Integer)
```

โดย X1,Y1 เป็นตำแหน่งมุมบนซ้าย และ
X2,Y2 เป็นตำแหน่งมุมล่างขวาของจุดที่ขยายออก

โพรซีเยอร์ Bar จะแสดงภาพสี่เหลี่ยมที่บริเวณ X1,Y1 และ X2,Y2 โดยมีลายอยู่ภายในเต็มพื้นที่

ลายที่โพรซีเยอร์ Bar แสดงในกรอบสี่เหลี่ยมเป็นลายที่กำหนดโดยโพรซีเยอร์ SetFillStyle ซึ่งมีรูปแบบดังนี้

```
SetFillStyle (Pattern : Word; Color : Word)
```

โดย Pattern คือลายที่กำหนดให้แก่ Bar ซึ่งมีค่าตั้งแต่ 0 ถึง 12 ในที่นี้จะกำหนดให้เป็น 1 (หรือ SolidFill - แถบสว่าง) สำหรับ Color ใช้กำหนดสีของภาพขยาย ในที่นี้กำหนดให้เป็น White (หรือ 15)

การขยายภาพจะมีลักษณะแตกต่างกันตามค่าที่กำหนดให้แก่ X1,Y1,X2,Y2 ในโพรซีเยอร์ Bar ซึ่งมีหลักสังเกตดังต่อไปนี้

ถ้า X2 - X1 = Y2 - Y1 ขยายภาพเท่ากันทั้งในทางกว้างและทางสูง
ถ้า X2 - X1 > Y2 - Y1 ขยายภาพในทางกว้างมากกว่าในทางสูง
ถ้า X2 - X1 < Y2 - Y1 ขยายภาพในทางสูงมากกว่าทางกว้าง
ถ้า X2 - X1 = 0 และ Y2 - Y1 > 0 ขยายภาพในทางสูง
ถ้า X2 - X1 > 0 และ Y2 - Y1 = 0 ขยายภาพในทางกว้าง

โพรซีเยอร์ขยายภาพคือ **ExpandPicture** ซึ่งมีรูปแบบดังนี้

```
ExpandPicture (X1, Y1, X2, Y2, X, Y,
               SizeX, SizeY : Integer)
```

โดย X1,Y1 คือตำแหน่งมุมซ้ายบนของภาพต้นแบบ

X2,Y2 คือตำแหน่งมุมขวาล่างของภาพต้นแบบ

X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

SizeX คือจำนวนเท่าของการขยายในทางกว้าง

SizeY คือจำนวนเท่าของการขยายในทางสูง

ตัวอย่างการขยายภาพแบบต่างๆ มีดังนี้

ตัวอย่าง โปรแกรม 7.11 แสดงการขยายภาพบนจอภาพ ซึ่งเป็นภาพที่ได้จากไฟล์ PUPPY.GTP โปรแกรมเริ่มด้วยการแสดงภาพขนาดปกติ เมื่อกดคีย์ Enter แต่ละครั้ง โปรแกรมจะแสดงภาพขยายในลักษณะต่างๆ ที่จอภาพรวม 3 ภาพ คือ ภาพขยายในทางสูง ภาพขยายในทางกว้าง และ ภาพขยายทั้งในทางสูงและทางกว้าง โดยกำหนดขนาดขยายเป็น 3 เท่า การเปลี่ยนค่า SizeX, SizeY จะมีผลต่อขนาดของภาพใหม่ และการเปลี่ยนพารามิเตอร์ตัวแรกของ SetFillStyle จะมีผลต่อลายของภาพใหม่

โปรแกรม 7.11

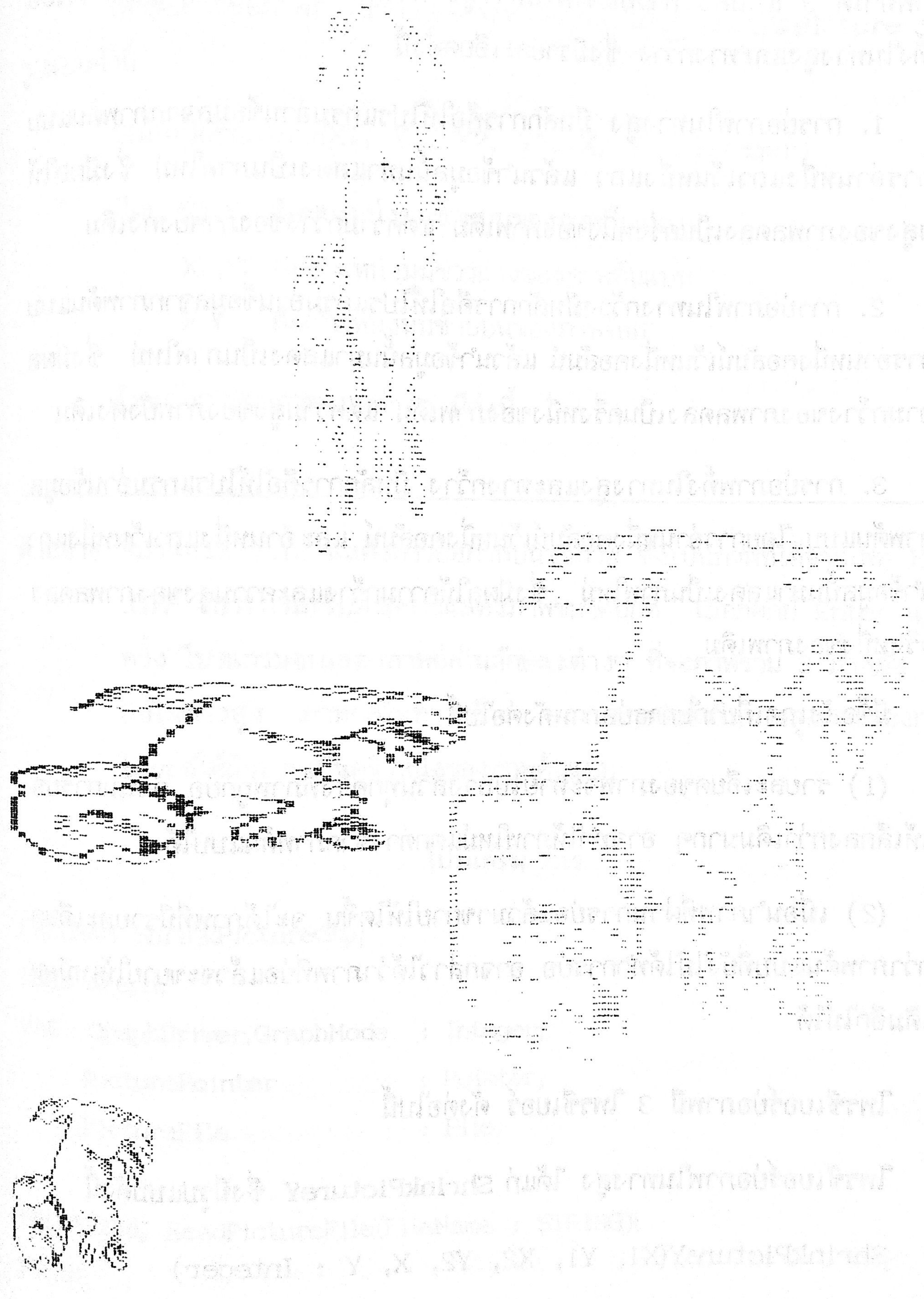
```
PROGRAM ExpandPictureFromGtp;
USES Graph;
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
  Assign(PictureFile,FileName);
  Reset(PictureFile);
```

```
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;

PROCEDURE ExpandPicture(X1,Y1,X2,Y2,X,Y,SizeX,SizeY : Integer);
VAR Row,Col,StartX,StartY : Integer;
BEGIN
   SetFillStyle(1,White);
   StartX := X; StartY := Y;
   FOR Row := Y1 TO Y2 DO
   BEGIN
      FOR Col := X1 TO X2 DO
      BEGIN
         IF GetPixel(Col,Row) <> 0 THEN Bar(X,Y,X,Y+(SizeY-1));
         Inc(X, SizeX);
      END;
      X := StartX; Inc(Y,SizeY);
   END;
END;

BEGIN
   ReadPictureFile('PUPPY.GTP');
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut); Readln;
   ExpandPicture(0,0,87,51,200,   0,1,3);     Readln;
   ExpandPicture(0,0,87,51,400,100,3,1);     Readln;
   ExpandPicture(0,0,87,51,100,200,3,3);     Readln;
   CloseGraph
END.
```



รูปที่ 7.14 ภาพบนจอภาพที่ได้จากโปรแกรม 7.11

7.7.2 การย่อภาพ หมายถึง การทำให้ภาพใหม่มีขนาดเล็กกว่าภาพเดิม การย่อภาพทำได้ 3 แบบคือ การย่อภาพในทางสูง การย่อภาพในทางกว้าง และการย่อภาพทั้งในทางสูงและทางกว้าง ซึ่งมีรายละเอียดดังนี้

1. การย่อภาพในทางสูง มีหลักการคือให้โปรแกรมอ่านข้อมูลจากภาพต้นแบบ โดยการอ่านหนึ่งแถวเว้นหนึ่งแถว แล้วนำข้อมูลนั้นมาแสดงเป็นภาพใหม่ ซึ่งมีผลให้ความสูงของภาพลดลงเป็นครึ่งหนึ่งของภาพเดิม แต่ความกว้างของภาพยังคงเดิม

2. การย่อภาพในทางกว้างมีหลักการคือให้โปรแกรมอ่านข้อมูลจากภาพต้นแบบ โดยการอ่านหนึ่งคอลัมน์เว้นหนึ่งคอลัมน์ แล้วนำข้อมูลนั้นมาแสดงเป็นภาพใหม่ ซึ่งมีผลให้ความกว้างของภาพลดลงเป็นครึ่งหนึ่งของภาพเดิม แต่ความสูงของภาพยังคงเดิม

3. การย่อภาพทั้งในทางสูงและทางกว้าง มีหลักการคือให้โปรแกรมอ่านข้อมูลจากภาพต้นแบบ โดยการอ่านหนึ่งคอลัมน์เว้นหนึ่งคอลัมน์ และ อ่านหนึ่งแถวเว้นหนึ่งแถว แล้วนำข้อมูลนั้นมาแสดงเป็นภาพใหม่ ซึ่งมีผลให้ความกว้างและความสูงของภาพลดลงเป็นครึ่งหนึ่งของภาพเดิม

มีข้อสังเกตเกี่ยวกับการย่อภาพดังต่อไปนี้

(1) รายละเอียดของภาพจะหายไปบางส่วนทุกครั้งที่ภาพถูกย่อ ดังนั้นการย่อภาพให้เล็กลงกว่าเดิมมากๆ อาจทำให้ภาพใหม่แตกต่างจากภาพต้นแบบได้

(2) เมื่อนำภาพที่ผ่านการย่อแล้วมาขยายให้โตขึ้น จะได้ภาพที่มีรายละเอียดน้อยกว่าภาพต้นแบบที่ยังไม่ได้ทำการย่อ อาจกล่าวได้ว่าภาพที่ย่อแล้วจะขยายให้เหมือนภาพเดิมอีกไม่ได้

ไพรซีเยอร์ย่อภาพมี 3 ไพรซีเยอร์ ดังต่อไปนี้

ไพรซีเยอร์ย่อภาพในทางสูง ได้แก่ **ShrinkPictureY** ซึ่งมีรูปแบบดังนี้

```
ShrinkPictureY(X1, Y1, X2, Y2, X, Y : Integer)
```

ไพรซีเยอร์ย่อภาพในทางกว้าง ได้แก่ **ShrinkPictureX** ซึ่งมีรูปแบบดังนี้

ShrinkPictureX(X1, Y1, X2, Y2, X, Y : Integer)

ไพรซีเยอร์ย่อภาพทั้งในทางสูงและทางกว้าง ได้แก่ **ShrinkPicture** ซึ่งมีรูปแบบดังนี้

ShrinkPicture(X1, Y1, X2, Y2, X, Y : Integer)

โดย X1,Y1 คือตำแหน่งมุมซ้ายบนของภาพต้นแบบ
X2,Y2 คือตำแหน่งมุมขวาล่างของภาพต้นแบบ
X,Y คือตำแหน่งมุมซ้ายบนของภาพใหม่

ตัวอย่างการย่อภาพแบบต่างๆ มีดังนี้

ตัวอย่าง โปรแกรม 7.12 แสดงการย่อภาพบนจอภาพ ซึ่งเป็นภาพที่ได้จากไฟล์ BIRD.GTP โปรแกรมเริ่มด้วยการแสดงภาพขนาดปกติ เมื่อกดคีย์ Enter แต่ละครั้ง โปรแกรมจะแสดงภาพย่อในลักษณะต่างๆ ที่จอภาพรวม 3 ภาพคือ ภาพย่อในทางสูง ภาพย่อในทางกว้าง และ ภาพย่อทั้งในทางสูงและทางกว้าง โดยย่อให้มีขนาดเป็นครึ่งหนึ่งของภาพต้นแบบ

โปรแกรม 7.12

```
PROGRAM ShrinkPictureGtp;
USES Graph;
VAR  GraphDriver,GraphMode  : Integer;
     PicturePointer         : Pointer;
     PictureFile            : File;

PROCEDURE ReadPictureFile(FileName : STRING);
BEGIN
  Assign(PictureFile,FileName);
```

```
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
END;

PROCEDURE ShrinkPictureX(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;
     Col,Row,StartX   : Integer;
BEGIN
   StartX := X;
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   FOR Row := Y1 TO Y2 DO
   BEGIN
      FOR Col := X1 TO X2 DO
      IF Col MOD 2 <> 0 THEN   {odd number row}
      BEGIN
         GetImage(Col,Row,Col,Row,PicturePointer^); {get a pixel}
         PutImage(X,Y,PicturePointer^,NormalPut);
         Inc(X);
      END;
      Inc(Y); X := StartX;
   END;
END;

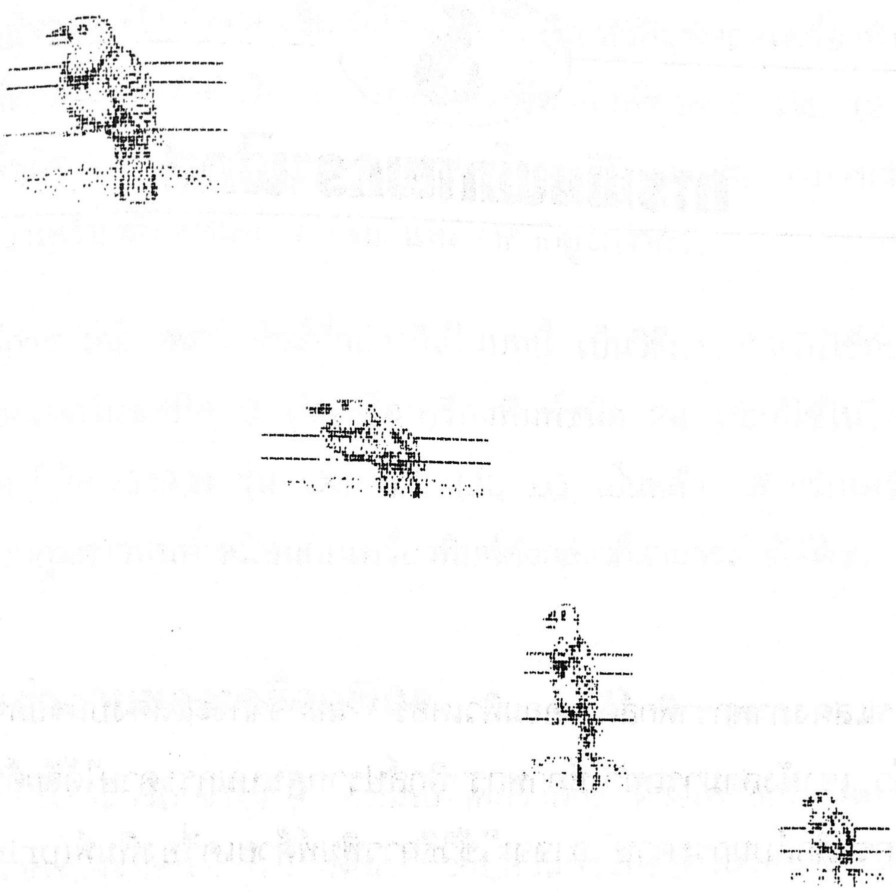
PROCEDURE ShrinkPictureY(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;
```

```
    Col,Row,StartX   : Integer;
BEGIN
   StartX := X;
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   FOR Row := Y1 TO Y2 DO
   BEGIN
      IF Row MOD 2 <> 0 THEN   {odd number row}
      BEGIN
         GetImage(X1,Row,X2,Row,PicturePointer^);
         PutImage(X,Y,PicturePointer^,NormalPut);
         Inc(Y);
      END;
      X := StartX;
   END;
END;

PROCEDURE ShrinkPicture(X1,Y1,X2,Y2,X,Y : Integer);
VAR  PicturePointer   : Pointer;
     MemUsed          : Word;
     Col,Row,StartX   : Integer;
BEGIN
   StartX := X;
   MemUsed := ImageSize(X1,Y1,X2,Y2);
   GetMem(PicturePointer,MemUsed);
   FOR Row := Y1 TO Y2 DO
   IF Row MOD 2 <> 0 THEN   {odd number row}
   BEGIN
      FOR Col := X1 TO X2 DO
```

```
      IF Col MOD 2 <> 0 THEN
      BEGIN
        GetImage(Col,Row,Col,Row,PicturePointer^);
        PutImage(X,Y,PicturePointer^,NormalPut);
        Inc(X);
      END;
      Inc(Y); X := StartX;
   END;
END;

BEGIN
   ReadPictureFile('BIRD.GTP');
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut); Readln;
   ShrinkPictureY(0, 0, 87, 51, 100,100);    Readln;
   ShrinkPictureX(0, 0, 87, 51, 200,150);    Readln;
   ShrinkPicture( 0, 0, 87, 51, 300,200);    Readln;
   CloseGraph
END.
```



รูปที่ 7.15 ภาพบนจอภาพที่ได้จากโปรแกรม 7.12

8

การพิมพ์ภาพกราฟิกส์

การแสดงภาพกราฟิกส์ด้วยคอมพิวเตอร์ นอกจากจะแสดงบนจอภาพของคอมพิวเตอร์แล้ว เรายังสามารถทำให้ภาพกราฟิกส์ปรากฏบนกระดาษได้อีกด้วย การทำให้เกิดภาพกราฟิกส์บนกระดาษ อาจจะใช้วิธีการพิมพ์ด้วยเครื่องพิมพ์(printer) หรือใช้วิธีการเขียนลงบนกระดาษด้วยพล็อตเตอร์(plotter) ก็ได้ เนื่องจากเราสามารถหาเครื่องพิมพ์มาใช้ได้ง่ายกว่าพล็อตเตอร์ เพราะมีการใช้เครื่องพิมพ์กันแพร่หลายกว่า ดังนั้นวิธีการสร้างภาพกราฟิกส์บนกระดาษที่จะกล่าวถึงในบทนี้ จึงเป็นวิธีการพิมพ์ด้วยเครื่องพิมพ์

8.1 ประเภทของเครื่องพิมพ์

เนื่องจากในปัจจุบันได้มีเครื่องพิมพ์อยู่หลายประเภท เช่น ดอตเมตริกซ์ (dot matrix printer) เลเซอร์ (laser printer) และเครื่องพิมพ์แต่ละประเภทยังมีอีกหลายรุ่น เครื่องพิมพ์แต่ละประเภท แต่ละรุ่น จะมีความแตกต่างกันมากบ้างน้อยบ้าง ดังนั้นการพิมพ์ภาพกราฟิกส์ด้วยเครื่องพิมพ์ต่างกันจึงมีวิธีการไม่เหมือนกัน

เมื่อพิจารณาถึงประเภทของเครื่องพิมพ์ที่มีใช้กันแพร่หลายในปัจจุบัน จะพบว่าส่วนใหญ่เป็นเครื่องพิมพ์ประเภทดอตเมตริกซ์ เครื่องพิมพ์ประเภทนี้มีส่วนดีที่ราคาไม่สูงมาก ใช้ง่าย ทนทาน มีประสิทธิภาพดี สามารถพิมพ์ได้ทั้งในโหมดเท็กซ์และโหมดกราฟิกส์

ถ้าพิจารณาถึงจำนวนเข็ม (pin) ที่มีอยู่ในหัวพิมพ์ของเครื่องพิมพ์ จะแบ่งเครื่องพิมพ์ดอตเมตริกซ์ได้เป็น 2 ชนิดคือ เครื่องพิมพ์ชนิด 9 เข็ม (9 pin) และเครื่องพิมพ์ชนิด 24 เข็ม (24 pin) เครื่องพิมพ์ชนิด 24 เข็ม สามารถพิมพ์รายละเอียดได้ดีกว่าเครื่องพิมพ์ชนิด 9 เข็ม และราคาก็สูงกว่าด้วย

วิธีการพิมพ์ภาพกราฟิกส์ที่กล่าวถึงในบทนี้ เป็นวิธีการสำหรับใช้กับเครื่องพิมพ์ประเภทดอตเมตริกซ์ชนิด 9 เข็มหรือเครื่องพิมพ์ชนิด 24 เข็มที่ใช้ในโหมด 9 เข็ม โดยใช้เครื่องยี่ห้อ EPSON รุ่น FX, MX, LX, LQ เป็นหลัก สำหรับเครื่องพิมพ์อื่นๆ ที่มีคำสั่งควบคุมการพิมพ์เหมือนกับเครื่องพิมพ์ดังกล่าวก็สามารถใช้ได้ด้วย

8.2 การทำงานของเครื่องพิมพ์

เพื่อให้สามารถเข้าใจวิธีการพิมพ์ภาพกราฟิกส์ ด้วยเครื่องพิมพ์ดอตเมตริกซ์ได้ง่ายขึ้น เราควรจะมีความรู้เบื้องต้นเกี่ยวกับส่วนประกอบและการทำงานของเครื่องพิมพ์ ดังต่อไปนี้

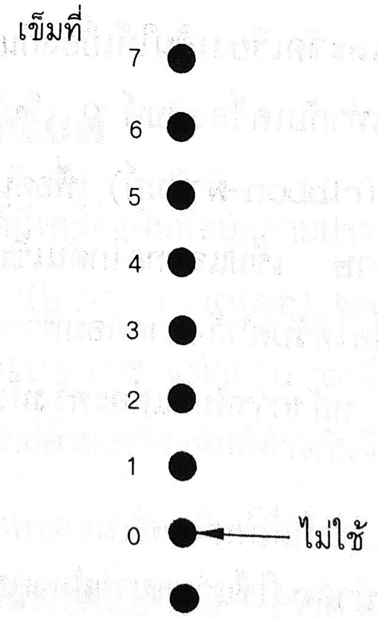
8.2.1 หัวพิมพ์ ส่วนประกอบสำคัญที่มีอยู่ในหัวพิมพ์คือ เข็มและกลไกดันเข็ม เข็มของเครื่องพิมพ์ 9 เข็มจัดเรียงเป็นเส้นตรงในแนวตั้ง สำหรับในเครื่องพิมพ์ 24 เข็มจะมีขนาดเข็มเล็กกว่า และจัดเรียงเข็มให้เยื้องกันเล็กน้อย เพื่อให้สามารถบรรจุเข็มทั้งหมดลงในเนื้อที่ขนาดเท่ากับเครื่องพิมพ์ 9 เข็มได้ เข็มแต่ละเข็มจะมีกลไกสำหรับดันเข็มออกไปที่ริบบอน (ribbon–ผ้าพิมพ์) เพื่อดันริบบอนให้ชนกับกระดาษ ทำให้เกิดรอยเข็มเป็นจุดขึ้นที่กระดาษ เข็มและกลไกดันเข็มแต่ละชุดจะทำงานแยกกันเป็นอิสระ เข็มจะถูกดันออกไปเมื่อได้รับคำสั่งจากคอมพิวเตอร์ การดันเข็มออกไปจะทำให้เกิดการพิมพ์ขึ้นบนกระดาษ หลังการพิมพ์แต่ละครั้งหัวพิมพ์จะเคลื่อนที่ตามแนวนอนไปยังตำแหน่งใหม่ที่จะพิมพ์

8.2.2 ภาพจากจุด ถ้าลองใช้แว่นขยายส่องดูภาพจากหนังสือพิมพ์ เราจะมองเห็นว่าภาพหนึ่งๆ ประกอบด้วยจุดเล็กๆ จำนวนมาก ปริมาณของจุดและตำแหน่งที่จุดเหล่านั้นปรากฏอยู่ทำให้เรามองเห็นเป็นภาพของสิ่งใดสิ่งหนึ่ง การพิมพ์ภาพกราฟิกส์

ด้วยเครื่องพิมพ์ดอตเมตริกซ์ก็อาศัยหลักการเดียวกันกับภาพในหนังสือพิมพ์คือ โดยการสั่งให้เครื่องพิมพ์ทำการพิมพ์จุดจำนวนต่างๆ ณ ตำแหน่งต่างๆ ที่กำหนดไว้บนกระดาษ ซึ่งจะทำให้มองเห็นเป็นภาพ เครื่องพิมพ์ 9 เข็มสามารถพิมพ์จุดในแนวนอนได้ถึง 240 จุด ต่อความยาวของกระดาษ 1 นิ้ว

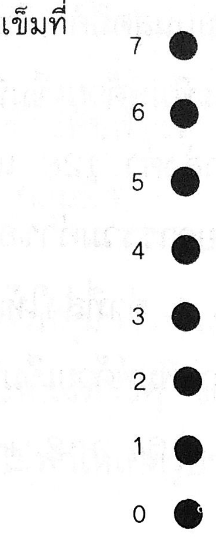
8.2.3 เครื่องพิมพ์ 9 เข็มและเครื่องพิมพ์ 24 เข็ม เนื่องจากเครื่องพิมพ์ชนิด 24 เข็ม (EPSON รุ่น LQ) สามารถทำงานเหมือนเครื่องพิมพ์ 9 เข็มได้ (คือใช้คำสั่งเหมือนกับเครื่อง 9 เข็มได้) ดังนั้นวิธีการพิมพ์ภาพด้วยเครื่องพิมพ์ 9 เข็มจึงนำมาใช้กับเครื่องพิมพ์ 24 เข็มได้ด้วย (แต่คำสั่งของเครื่องพิมพ์ 24 เข็มจะใช้กับเครื่องพิมพ์ 9 เข็มไม่ได้) เพราะฉะนั้นโปรแกรมพิมพ์ภาพในบทนี้จึงสามารถใช้กับเครื่องพิมพ์รุ่น LQ ได้ด้วย

8.2.4 จำนวนเข็มที่ใช้พิมพ์ภาพกราฟิกส์ ในการพิมพ์ภาพกราฟิกส์ เครื่องพิมพ์จะใช้เข็มเพียง 8 เข็มเท่านั้น เครื่องพิมพ์ชนิด 9 เข็มจะใช้ 8 เข็มบนเว้นเข็มล่างสุด 1 เข็ม เครื่องพิมพ์ 24 เข็มจะใช้ 8 เข็มจากชุดหนึ่ง อีก 2 ชุดที่เหลือไม่ได้ใช้ ดังแสดงในรูปที่ 8.1



(ก) เครื่องพิมพ์ 9 เข็ม

รูปที่ 8.1 เข็มของเครื่องพิมพ์ชนิด 9 เข็มและ 24 เข็มที่ใช้พิมพ์ภาพกราฟิกส์

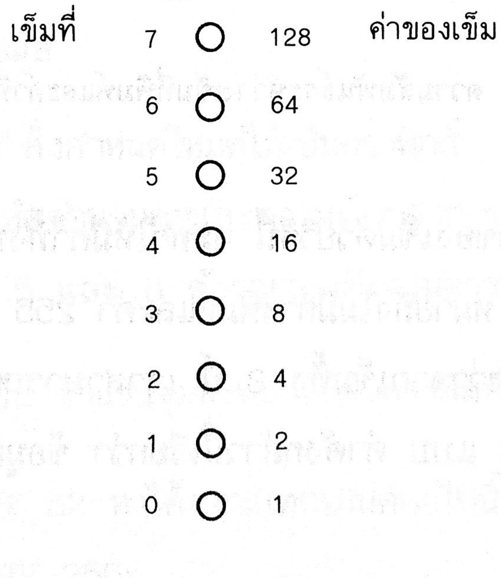


(ข) เครื่องพิมพ์ 24 เข็ม

รูปที่ 8.1 (ต่อ) เข็มของเครื่องพิมพ์ชนิด 9 เข็มและ 24 เข็มที่ใช้พิมพ์ภาพกราฟฟิกส์

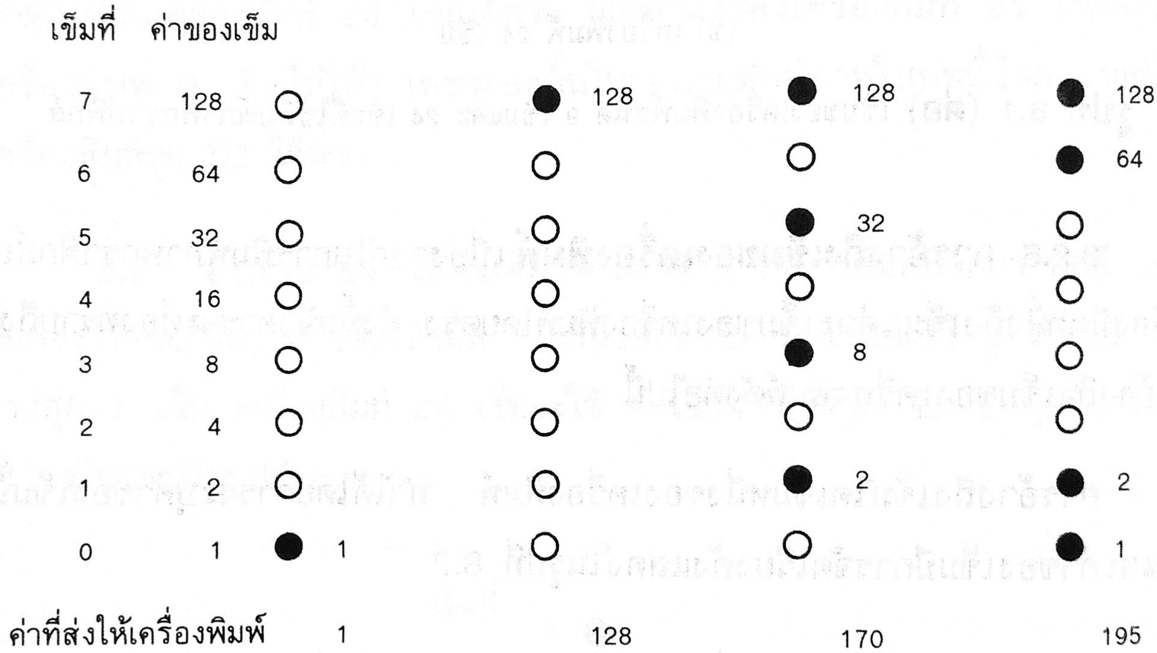
8.2.5 การอ้างถึงเข็มของเครื่องพิมพ์ เนื่องจากในการพิมพ์ภาพกราฟิกส์เราจะต้องมีคำสั่งถึงเข็มแต่ละเข็มของเครื่องพิมพ์โดยตรง ดังนั้นจึงควรจะต้องทราบถึงวิธีการอ้างถึงเข็มของเครื่องพิมพ์ดังต่อไปนี้

การอ้างถึงเข็มใดเข็มหนึ่งของเครื่องพิมพ์ ทำได้โดยการระบุค่าของเข็มนั้นเข็มและค่าของเข็มมีการจัดเรียงดังแสดงในรูปที่ 8.2



รูปที่ 8.2 ตำแหน่งเข็มและค่าของแต่ละเข็ม

เข็มล่างสุดมีค่าเป็น 1 และเข็มบนสุดมีค่าเป็น 128 ค่าเหล่านี้กำหนดไว้เพื่อส่งไปให้เครื่องพิมพ์ โดยเครื่องพิมพ์จะพิมพ์ด้วยเข็มที่มีค่าตามที่ระบุไว้ เช่น ถ้าส่งค่า 1 เครื่องจะพิมพ์ด้วยเข็มล่างสุด ถ้าส่งค่า 128 เครื่องจะพิมพ์ด้วยเข็มบนสุด สำหรับการพิมพ์ครั้งละหลายเข็ม ทำได้โดยการรวมค่าของแต่ละเข็มเข้าด้วยกัน เช่น ถ้าต้องการพิมพ์ด้วยเข็มที่ 7, 5, 3 และ 1 ค่าที่ส่งให้แก่เครื่องพิมพ์คือ 170 ซึ่งได้จาก 128 + 32 + 8 + 2 ถ้าต้องการพิมพ์ด้วยเข็มที่ 7, 6, 1 และ 0 ค่าที่ส่งให้แก่เครื่องพิมพ์คือ 195 ซึ่งได้จาก 128 + 64 + 2 + 1 ดังแสดงในรูปที่ 8.3



รูปที่ 8.3 ความสัมพันธ์ระหว่างเข็มที่พิมพ์และค่าที่ส่งให้แก่เครื่องพิมพ์

การกำหนดค่าของเข็มด้วยวิธีนี้ จะทำให้มีค่าทั้งหมดรวม 256 ค่า จากค่า 0 ถึง 255 โดยค่า 0 หมายถึงไม่มีการพิมพ์และค่า 255 หมายถึงพิมพ์ทุกเข็ม ซึ่งจะได้ภาพเป็นเส้นตั้ง แสดงว่าจากเข็มทั้ง 8 นี้ เราสามารถกำหนดรูปแบบการพิมพ์ให้แตกต่างกันได้ไม่เกิน 256 แบบ ค่าดังกล่าวนี้เรียกว่า ข้อมูล (data) ของเครื่องพิมพ์

8.3 คำสั่งพิมพ์กราฟิกส์

เราได้ทราบมาแล้วว่าเครื่องพิมพ์จะทำตามคำสั่งที่ได้รับจากคอมพิวเตอร์ คำสั่งที่ใช้กับเครื่องพิมพ์แบ่งเป็น 2 กลุ่มคือ

1. *คำสั่งควบคุม* ใช้ควบคุมการทำงานบางอย่างของเครื่องพิมพ์ เช่น ขึ้นบรรทัดใหม่เลื่อนหัวพิมพ์ไปที่ตำแหน่งต่างๆ ของบรรทัด จัดระยะห่างระหว่างบรรทัดและขึ้นหน้าใหม่ คำสั่งในกลุ่มนี้ไม่ทำให้เกิดรูปรอยใดๆ ที่กระดาษ

2. *คำสั่งพิมพ์* เป็นคำสั่งที่ทำให้เกิดรูปรอยขึ้นบนกระดาษ คำสั่งพิมพ์แบ่งเป็น 2 ประเภทคือ

ก. คำสั่งพิมพ์ข้อความใช้สำหรับพิมพ์อักษรต่างๆตามที่กำหนดไว้ในรหัส ASCII

ข. คำสั่งพิมพ์กราฟิกส์ใช้สำหรับพิมพ์ให้เป็นภาพต่างๆ ตามที่เราต้องการ

การสั่งเครื่องพิมพ์ให้พิมพ์กราฟิกส์จะต้องใช้คำสั่งอย่างน้อย 3 คำสั่งดังต่อไปนี้

8.3.1 คำสั่งกำหนดโหมดพิมพ์กราฟิกส์ ตามปกติเครื่องพิมพ์จะทำงานในโหมดเท็กซ์ซึ่งใช้สำหรับการพิมพ์ข้อความ ถ้าต้องการให้พิมพ์ภาพกราฟิกส์ เราจะต้องใช้คำสั่งเปลี่ยนโหมดพิมพ์ ซึ่งมีรูปแบบของคำสั่งดังนี้

ESC ⋆ m n1 n2 ข้อมูล

โดย ESC ⋆ เป็นคำสั่งกำหนดโหมดให้เป็นกราฟิกส์

m เป็นรหัสจำนวนจุดและความเร็วของการพิมพ์มีค่าเป็น 0 1 2 3 4 5 และ 6 ดังรายละเอียดในตารางที่ 8.1

n1 และ n2 เป็น จำนวนจุดทั้งหมดที่ต้องการพิมพ์ใน 1 แถว

ค่าของ n1 และ n2 หาได้จากสเตตเมนต์ต่อไปนี้

```
n2  :=  N DIV 256;
n1  :=  N MOD 256
```

ตารางที่ 8.1 ค่า m จำนวนจุด ระยะห่างของจุดและความเร็วของการพิมพ์

m	จำนวนจุดในแนวนอน (จุดต่อนิ้ว)	ระยะห่างแต่ละจุด (นิ้ว)	ความหมาย
0	60	0.0167	จำนวนจุดมาตรฐาน
1	120	0.0083	จำนวนจุด 2 เท่า
2	120	0.0083	จำนวนจุด 2 เท่า พิมพ์เร็ว
3	240	0.0042	จำนวนจุด 4 เท่า
4	80	0.0125	จำนวนจุด 80 จุดต่อนิ้ว
5	72	0.0133	จำนวนจุด 72 จุดต่อนิ้ว
6	90	0.0111	จำนวนจุด 90 จุดต่อนิ้ว

หรือ

```
n2  :=  Int(N / 256);
n1  :=  N - (256 * n2)
```

โดย N คือจำนวนจุดที่ต้องการพิมพ์ใน 1 บรรทัด

Int คือฟังก์ชันหาค่าจำนวนเต็มจาก N/256

ดังนั้น ค่า n2 ก็คือจำนวนเต็มที่ได้จากการหารค่าจำนวนจุดใน 1 แถว ด้วย 256 และ n1 ก็คือเศษที่เหลือจากการหารดังกล่าว

ตัวอย่างที่ 1 สมมติว่าเราต้องการพิมพ์จุด บรรทัดละ 100 จุด จะหาค่า n2 และ n1 ได้ดังนี้

n2 = 100 DIV 256 = 0 (ผลลัพธ์เฉพาะจำนวนเต็ม)

n1 = 100 MOD 256 = 100 (เศษที่เหลือ)

ตัวอย่างที่ 2 สมมติว่าเราต้องการพิมพ์จุด บรรทัดละ 967 จุด จะหาค่า n2 และ n1 ได้ดังนี้

n2 = 967 / 256 = 3 (ผลลัพธ์เฉพาะจำนวนเต็ม)

n1 = 967 - (256 * 3) = 199 (เศษที่เหลือ)

จะเห็นได้ว่า ถ้าจุดที่ต้องการพิมพ์มีจำนวนน้อยกว่า 256 ค่า n2 จะเป็น 0 และ n1 จะมีค่าเท่ากับจำนวนจุดที่ต้องการพิมพ์พอดี กรณีนี้จึงไม่จำเป็นต้องคำนวณ การคำนวณหาค่า n1 และ n2 จะต้องทำก็ต่อเมื่อจำนวนจุดที่ต้องการพิมพ์มีค่าตั้งแต่ 256 ขึ้นไปเท่านั้น

รูปแบบของคำสั่ง ESC * m n1 n2 ข้อมูล เขียนเป็นรหัส ASCII ได้ดังนี้

รหัสเลขฐานสิบ 27 42 m n1 n2 ข้อมูล

รหัสเลขฐานสิบหก 1B 2A m n1 n2 ข้อมูล

ตัวอย่างที่ 3 สมมติว่าเราต้องการสั่งให้เครื่องพิมพ์เปลี่ยนโหมดเป็นกราฟิกส์ ที่มีจำนวนจุดในแนวนอนเท่ากับ 120 จุดต่อนิ้ว พิมพ์บรรทัดละ 100 จุด จะเขียนสเตตเมนต์ได้ดังนี้

```
Write (Lst, #27#42#2#100#0)
```

และ

```
Write (Lst, #$1B#$2A#$2#$64#$0)
```

โดย Lst เป็นแวเรียเบิลชนิดเท็กซ์ไฟล์ซึ่งหมายถึงเครื่องพิมพ์ Lst อยู่ในยูนิต Printer

8.3.2 คำสั่งขึ้นบรรทัดใหม่ เราทราบแล้วว่า เครื่องพิมพ์จะพิมพ์เป็นรูปร่างต่างๆ ตามข้อมูลที่เข็มของเครื่องพิมพ์ได้รับมาจากคอมพิวเตอร์ เมื่อได้รับข้อมูล 1 ค่า เครื่องพิมพ์จะพิมพ์ 1 ครั้งเนื่องจากการรับส่งข้อมูลระหว่างเครื่องพิมพ์กับคอมพิวเตอร์มีความเร็วสูงมาก เครื่องพิมพ์จึงพิมพ์ต่อเนื่องกันไปจนหมดข้อมูล อย่างไรก็ตามเมื่อหัว

พิมพ์เคลื่อนมาถึงตำแหน่งขวาสุด (หรือซ้ายสุด) แล้ว หัวพิมพ์จะเคลื่อนที่ต่อไปไม่ได้อีก เครื่องพิมพ์จะเลื่อนกระดาษขึ้นมาแทนที่อีก 1 บรรทัดเรียกว่า *การขึ้นบรรทัดใหม่* นอกจากการขึ้นบรรทัดใหม่แบบอัตโนมัติดังกล่าวนี้แล้ว เรายังสามารถสั่งให้เครื่องพิมพ์ขึ้นบรรทัดใหม่ ณ ตำแหน่งที่เราต้องการเองก็ได้ โดยใช้คำสั่งขึ้นบรรทัดใหม่ ซึ่งมีรูปแบบของคำสั่งดังนี้

เลขฐานสิบ	10
เลขฐานสิบหก	0A

คำสั่งนี้จะมีผลให้เครื่องพิมพ์นำข้อมูลที่มีอยู่ในหน่วยความจำของเครื่องพิมพ์ออกมาพิมพ์ที่กระดาษ แล้วเลื่อนกระดาษขึ้น 1 บรรทัด แต่ถ้าไม่มีข้อมูลอยู่ในหน่วยความจำของเครื่องพิมพ์ จะมีการเลื่อนกระดาษขึ้น 1 บรรทัดเพียงอย่างเดียวโดยไม่มีการพิมพ์ใดๆ

การใช้คำสั่งขึ้นบรรทัดใหม่ มีตัวอย่างสเตตเมนต์ ดังนี้

```
Write (Lst, #10)
```

หรือ

```
Write (Lst, #$0A)
```

สเตตเมนต์ดังกล่าวนี้จะทำงานเหมือนกับสเตตเมนต์ต่อไปนี้

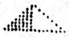
```
WriteLn (Lst)
```

ตัวอย่าง โปรแกรม 8.1 ใช้สำหรับการพิมพ์จุดที่จัดเรียงจาก 1 จุดจนถึง 7 จุดเพื่อให้มองเห็นเป็นภาพสามเหลี่ยมซึ่งครึ่งหนึ่งโปร่งและอีกครึ่งหนึ่งทึบ

โปรแกรม 8.1

```
PROGRAM GrTri;
USES Printer;
```

```
BEGIN
   Write(Lst,#27#42#0#16#0); {set 60 dot/inch,16 column}
   Write(Lst,#1#3#7#15#31#63#127#255#129#65#33#17#9#5#3#1);
   Write(Lst,#10);     {line feed}
END.
```

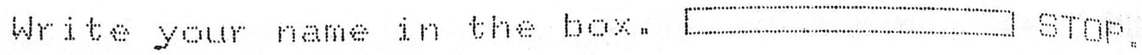


รูปที่ 8.4 ภาพจากโปรแกรม 8.1

ตัวอย่าง โปรแกรม 8.2 ใช้สำหรับพิมพ์ฟอร์มเพื่อให้คนกรอกชื่อลงในกรอบที่กำหนด โปรแกรมนี้จะทำงานในโหมดเท็กซ์เพื่อพิมพ์ข้อความและทำงานในโหมดกราฟิกส์เพื่อพิมพ์กรอบ

โปรแกรม 8.2

```
PROGRAM GrBox;
USES Printer;
VAR Col : Integer;
BEGIN
   Write(Lst,'Write your name in the box. '); {print text}
   Write(Lst,#27#42#2#82#0); {120 dot/inch,high speed,82 column}
   Write(Lst,#255);              {left side}
   FOR Col := 1 TO 80 DO Write(Lst,#129); {top & bottom line}
   Write(Lst,#255);              {right side}
   Write(Lst,' STOP.');          {print text}
   Write(Lst,#10);               {line feed}
END.
```



รูปที่ 8.5 ภาพจากโปรแกรม 8.2

8.3.3 คำสั่งตั้งระยะห่างระหว่างบรรทัด ในกรณีที่มีข้อมูลเกินกว่า 1 บรรทัด เมื่อเครื่องพิมพ์ พิมพ์บรรทัดแรกเสร็จแล้ว กระดาษจะถูกเลื่อนขึ้นมาเป็นการขึ้นบรรทัดใหม่ ตามปกติเครื่องพิมพ์จะขึ้นบรรทัดใหม่โดยมีระยะห่างระหว่างบรรทัดเท่ากับ 1/6 นิ้ว แต่เราสามารถเปลี่ยนค่านี้ได้ ด้วยคำสั่งที่มีรูปแบบดังต่อไปนี้

ESC 3 n

คำสั่งนี้ใช้สำหรับตั้งระยะห่างระหว่างบรรทัดให้มีค่าเท่ากับ n/216 นิ้วสำหรับเครื่องพิมพ์ 9 เข็ม และเท่ากับ n/180 นิ้วสำหรับเครื่องพิมพ์ 24 เข็ม โดย n มีค่าตั้งแต่ 0 ถึง 255

คำสั่งตั้งระยะห่างระหว่างบรรทัด จะใช้ในกรณีที่ภาพมีความสูงเกินกว่า 1 บรรทัด (สูงกว่า 8 จุด) ซึ่งเมื่อเครื่องพิมพ์ทำการพิมพ์บรรทัดแรกเสร็จแล้ว กระดาษจะถูกเลื่อนขึ้นมาเพื่อขึ้นบรรทัดใหม่ ถ้าใช้ระยะห่างระหว่างบรรทัดตามปกติคือ 1/6 นิ้ว จะเกิดความไม่เหมาะสมกับการพิมพ์ภาพกราฟิกส์ เพราะจะมีแถวว่างอยู่ระหว่างบรรทัดที่มีการพิมพ์ ทำให้เห็นภาพถูกตัดออกเป็นริ้ว วิธีการแก้ไขเพื่อให้มีการพิมพ์ภาพต่อกันเป็นภาพเดียวทำได้โดยการจัดระยะห่างระหว่างบรรทัดให้น้อยลงกว่าปกติ ซึ่งกำหนดด้วยค่าของ n วิธีหาค่าของ n มีดังต่อไปนี้

จากหนังสือคู่มือการใช้เครื่องพิมพ์ เราทราบว่าความสูงของอักษรซึ่งมี 8 จุด เท่ากับ 0.095 นิ้ว

ดังนั้นระยะห่างแต่ละจุดจึงเท่ากับ 0.095 / 8 = 0.01188 นิ้ว

ระยะบรรทัดที่ต้องการ = ความสูง 8 จุด + ระยะห่าง 1 จุด

= 0.095 + 0.01188

= 0.10688 นิ้ว

คำนวณค่า n จากสมการต่อไปนี้

ระยะบรรทัดที่ต้องการ = n / 216

0.10688 = n / 216

n = 0.10688 x 216

= 23.086

เนื่องจากค่า n ต้องเป็นจำนวนเต็มดังนั้นค่า n ที่เหมาะสมจึงควรเท่ากับ 23

สเตตเมนต์ตั้งระยะห่างระหว่างบรรทัดเป็นดังนี้

```
Write (Lst, #27#51#23)
```

หรือ

```
Write (Lst, #$1B#$33#$17)
```

นอกจากคำสั่งดังกล่าวแล้วนี้ ก่อนจบโปรแกรมพิมพ์ภาพกราฟิกส์ เราควรจะเพิ่มเติมคำสั่งอีก 2 คำสั่งคือ

1. คำสั่งเลื่อนกระดาษขึ้นหน้าใหม่ ซึ่งจะทำให้กระดาษที่พิมพ์เสร็จแล้วหลุดออกมาจากเครื่องพิมพ์

สเตตเมนต์เลื่อนกระดาษขึ้นหน้าใหม่เป็นดังนี้

```
Write (Lst, #12)
```

หรือ

```
Write (Lst, #$0C)
```

2. คำสั่งตั้งระยะห่างหว่างบรรทัดให้มีค่าเป็นปกติคือ 1/6 นิ้ว รูปแบบของคำสั่งเป็นดังนี้

ESC 2

สเตตเมนต์ตั้งระยะห่างระหว่างบรรทัดขนาด 1/6 นิ้ว เป็นดังนี้

```
Write (Lst, #50)
```

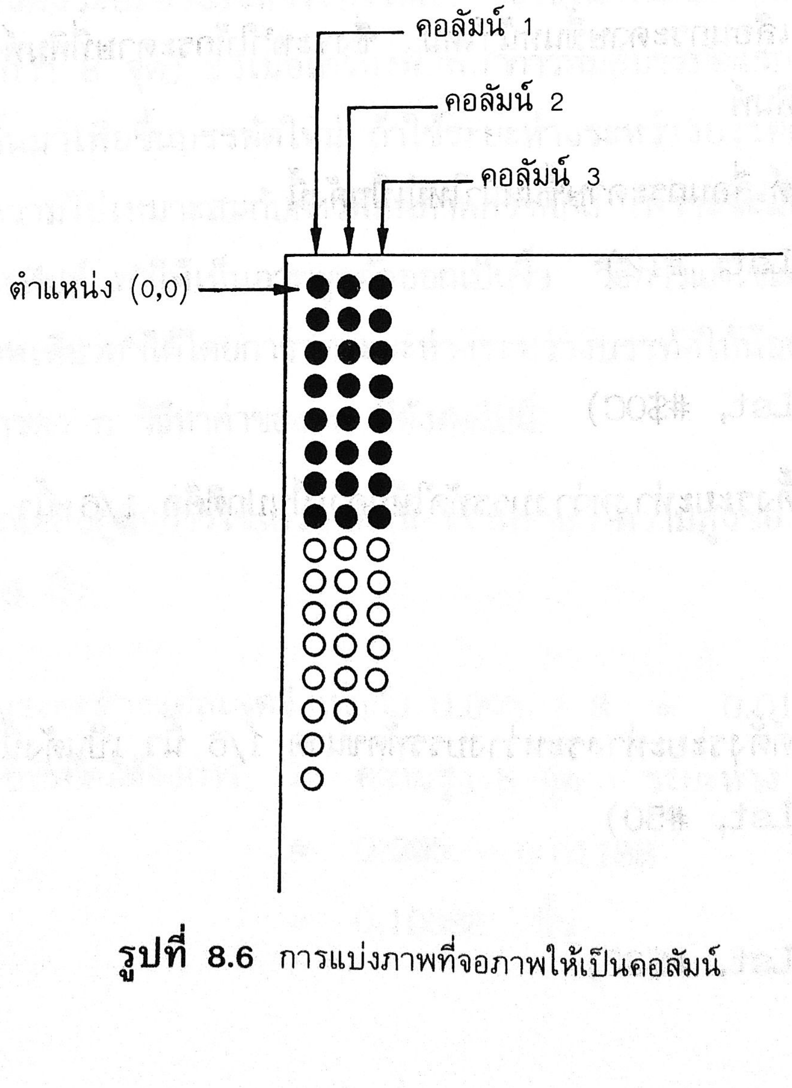
หรือ

```
Write (Lst, #$32)
```

8.4 การพิมพ์ภาพกราฟิกส์จากจอภาพ

เราทราบว่าภาพที่ปรากฏบนจอภาพ และภาพที่ปรากฏบนกระดาษเป็นภาพที่เกิดขึ้นโดยหลักการเดียวกันคือ ทำให้เกิดจุดจำนวนต่างๆ อยู่ ณ ตำแหน่งต่างๆกันที่จอภาพหรือบนกระดาษแล้วแต่กรณี ดังนั้นการนำภาพจากจอภาพไปพิมพ์บนกระดาษจึงสามารถทำได้ โดยใช้หลักการง่ายๆ คือ ทำการก๊อบปี้จุดจากจอภาพไปพิมพ์บนกระดาษ เนื่องจากจอภาพแต่ละชนิดมีจำนวนจุดไม่เท่ากัน จึงมีผลให้โปรแกรมสำหรับพิมพ์ภาพด้วยหลักการนี้มีรายละเอียดแตกต่างกันบ้างเล็กน้อย อย่างไรก็ดี หลักการทำงานของโปรแกรมก็ยังคงเหมือนกัน ในที่นี้จะขอยกตัวอย่างโปรแกรมพิมพ์ภาพจากจอภาพสีเดียว ที่มีระบบกราฟิกส์เป็นเฮอร์คิวลีส ซึ่งมีวิธีการตามขั้นตอนต่อไปนี้

8.4.1 แบ่งจอภาพออกเป็นคอลัมน์เรียงจากซ้ายไปขวา ให้แต่ละคอลัมน์มี 8 จุดเรียงจากบนลงล่าง ดังแสดงในรูปที่ 8.6



รูปที่ 8.6 การแบ่งภาพที่จอภาพให้เป็นคอลัมน์

การแบ่งจอภาพตามหลักการนี้ จะทำให้มีคอลัมน์แรกอยู่ที่ด้านซ้ายสุดของจอภาพ โดยจุดบนสุดของคอลัมน์เป็นจุดที่ตำแหน่ง 0,0 ของจอภาพ จุดที่ 2 คือตำแหน่ง 0,1 จุดที่ 3 คือตำแหน่ง 0,2 เรียงกันตามลำดับถึงตำแหน่ง 0,7 แสดงตัวอย่างความสัมพันธ์ระหว่างคอลัมน์และตำแหน่งของจุดบนจอภาพดังรูปที่ 8.7

	คอลัมน์ 1	คอลัมน์ 2	. . .	คอลัมน์ 720
	X,Y	X,Y	. . .	X,Y
	0,0	1,0	. . .	719,0
	0,1	1,1	. . .	719,1
	0,2	1,2	. . .	719,2
แถบที่ 1	0,3	1,3	. . .	719,3
	0,4	1,4	. . .	719,4
	0,5	1,5	. . .	719,5
	0,6	1,6	. . .	719,6
	0,7	1,7	. . .	719,7

รูปที่ 8.7 คอลัมน์และตำแหน่งจุดบนจอภาพแบบเฮอร์คิวลิส

เนื่องจากระบบกราฟิกส์แบบเฮอร์คิวลีสกำหนดจำนวน คอลัมน์ x แถว เป็น 720 x 348 ดังนั้นจึงแบ่งเป็นคอลัมน์ได้ 720 คอลัมน์ สำหรับในแนวนอนซึ่งประกอบด้วย 348 แถว จะถูกจัดแบ่งเป็นแถบ (band) ให้หนึ่งแถบมี 8 แถว (คือ 8 จุดของ 1 คอลัมน์) ดังนั้นจึงหาจำนวนแถบได้จาก 348/8 = 43.5 ซึ่งจะต้องกำหนดให้เป็น 44 แถบเพื่อให้ได้ภาพครบทุกแถว

สรุปได้ว่า ในการพิมพ์ภาพเต็มจอภาพ จากระบบกราฟิกส์แบบเฮอร์คิวลีสนั้น เครื่องพิมพ์จะพิมพ์ 44 บรรทัด (บรรทัดละ 1 แถบหรือ 8 แถว) แต่ละบรรทัดมี 720 คอลัมน์ แต่ละคอลัมน์มีการพิมพ์ 1 ครั้ง ดังนั้นจำนวนครั้งของการพิมพ์ต่อหนึ่งจอภาพจึงเท่ากับ 720 x 44 = 31680 ครั้ง

เทอร์โบปาสคาลมีฟังก์ชันสำหรับหาค่าคอลัมน์ และแถวที่มีอยู่ในระบบกราฟิกส์แต่ละระบบคือ

GetMaxX

ฟังก์ชันจะให้ค่าคอลัมน์ขวาสุดของระบบกราฟิกส์นั้น โดยเริ่มค่า 0 จากทางซ้ายสุด เช่น ระบบกราฟิกส์เฮอร์คิวลีสจะมีค่า GetMaxX เป็น 719 ดังนั้นจำนวนคอลัมน์ของกราฟิกส์แต่ละระบบจึงเท่ากับ GetMaxX + 1

GetMaxY

ฟังก์ชันจะให้ค่าแถวล่างสุดของระบบกราฟิกส์นั้น โดยเริ่มค่า 0 จากแถวบนสุด เช่น ระบบกราฟิกส์เฮอร์คิวลีสจะมีค่า GetMaxY เป็น 347 ดังนั้นจำนวนแถวของกราฟิกส์แต่ละระบบจึงเท่ากับ GetMaxY + 1

เพราะฉะนั้น การเขียนโปรแกรมเพื่อให้สามารถนำไปใช้ได้กับระบบกราฟิกส์ทุกระบบเราจึงควรใช้ฟังก์ชันทั้งสองนี้

8.4.2 ตรวจการแสดงจุดที่จอภาพในคอลัมน์ โดยการตรวจทุกจุดตั้งแต่จุด 0 ถึงจุดที่ 7 ด้วยฟังก์ชัน **GetPixel** ซึ่งมีรูปแบบดังนี้

GetPixel (X, Y : Integer)

โดย X, Y คือตำแหน่งที่ต้องการตรวจจุด

ค่าจากฟังก์ชันจะเป็นตัวเลข 0 - 15 ซึ่งหมายถึงสีต่างๆ ดังนี้

0 = Black
1 = Blue
2 = Green
3 = Cyan
8 = DarkGray
9 = LightBlue
10 = LightGreen
11 = LightCyan

4	=	Red	12	=	LightRed
5	=	Magenta	13	=	LightMagenta
6	=	Brown	14	=	Yellow
7	=	LightGray	15	=	White

ตามปกติการสั่งแสดงจุดที่จอภาพ ถ้าต้องการให้มองเห็นจุดนั้นเราจะต้องสั่งให้แสดงจุดด้วยสีที่ไม่ใช่สีพื้น เช่น ถ้าสีพื้นเป็นสีดำแล้วเราสั่งแสดงจุดด้วยสีดำเราจะมองไม่เห็นจุดนั้นเนื่องจากเป็นสีเดียวกัน และเราจะถือว่าไม่มีการแสดงจุด ณ ตำแหน่งนั้น เพราะฉะนั้นเราจึงตรวจสอบการแสดงจุดบนจอภาพ ณ จุด X,Y ใดๆได้โดยการตรวจว่าค่าสีของจุดนั้นเท่ากับค่าของสีพื้นหรือไม่ ถ้าเท่ากันแสดงว่าไม่มีการแสดงจุดที่ X,Y นั้น แต่ถ้าค่าไม่เท่ากัน แสดงว่ามีการแสดงจุดที่ X,Y นั้น

วิธีตรวจการแสดงจุดทำได้ด้วยฟังก์ชัน **GetPixel** และ **GetBkColor** ซึ่งมีรูปแบบดังต่อไปนี้

GetPixel (X, Y)

ฟังก์ชันจะให้ค่าตัวเลข 0 ถึง 15 ซึ่งเป็นค่าสีของจุด ณ ตำแหน่ง X,Y

GetBkColor

ฟังก์ชันจะให้ค่าตัวเลข 0 ถึง 15 ซึ่งเป็นค่าของสีที่ใช้เป็นสีพื้นอยู่ในขณะนั้น

สเตตเมนต์ตรวจสอบการแสดงจุดเป็นดังนี้

IF GetPixel (X, Y) <> GetBkColor {ถ้าจุดนี้ไม่ใช่สีพื้น}

ถ้าผลลัพธ์เป็นจริง หมายถึงมีจุดแสดงที่ X,Y

แต่ถ้าผลลัพธ์เป็นเท็จ หมายถึงไม่มีจุดแสดงที่ X,Y

8.4.3 เปลี่ยนค่าจากคอลัมน์ให้เป็นข้อมูลสำหรับพิมพ์ โดยมีเงื่อนไขว่าถ้าไม่แสดงจุดให้มีค่าเป็น 0 ถ้าแสดงจุดให้มีค่าต่างๆ ตามแถวของจุดดังรูปที่ 8.8

จุดแถวที่	มีค่า
0	128
1	64
2	32
3	16
4	8
5	4
6	2
7	1

รูปที่ 8.8 ค่าของจุดในแถวต่าง ๆ ที่มีการแสดงจุด

ค่าข้อมูลสำหรับพิมพ์หาได้จากการรวมค่าของจุดทั้งหมดเข้าด้วยกัน ซึ่งจะมีค่าตั้งแต่ 0 ถึง 255 การหาค่าข้อมูลสำหรับพิมพ์มีวิธีการเช่นเดียวกับที่ได้กล่าวแล้วในข้อ 8.2.5 สำหรับความแตกต่างของการกำหนดตำแหน่งเข็มของเครื่องพิมพ์กับตำแหน่งแถวของจุดบนจอภาพจะไม่ทำให้ข้อมูลที่ส่งให้เครื่องพิมพ์มีค่าต่างกัน เนื่องจากแต่ละตำแหน่งมีค่าเท่ากัน และมีการเรียงลำดับจากค่ามากลงมาหาค่าน้อยเช่นเดียวกัน

8.4.4 สั่งเครื่องพิมพ์ให้ทำงาน ซึ่งมีรายการที่ต้องทำดังต่อไปนี้

1. จัดระยะห่างระหว่างบรรทัดเป็น 23/216 ด้วยสเตตเมนต์ต่อไปนี้

```
Write(Lst,#27#51#23)
```

2. เปลี่ยนโหมดเป็นกราฟิกส์ โดยกำหนดให้พิมพ์ 90 จุดต่อนิ้ว ซึ่งจะได้ภาพที่มีสัดส่วนใกล้เคียงกับจอภาพมากที่สุด สเตตเมนต์ที่ใช้คือ

```
Write(Lst,#27#42#6+Chr(N1)+Chr(N2))
```

สำหรับค่า N1 และ N2 หาได้จากสเตตเมนต์ต่อไปนี้

```
N2 := GetMaxX DIV 256;
N1 := GetMaxX MOD 256
```

3. รับข้อมูลใหม่มาจากคอมพิวเตอร์เพื่อพิมพ์ 1 คอลัมน์

4. เมื่อพิมพ์ครบตามคอลัมน์ที่กำหนด ให้ขึ้นบรรทัดใหม่

5. เมื่อพิมพ์ครบทุกบรรทัด ให้หยุดพิมพ์

6. เลื่อนกระดาษขึ้น 1 หน้าและจัดระยะห่างระหว่างบรรทัดเป็น 1/16 ด้วยสเตตเมนต์ต่อไปนี้

```
Write(Lst,#12#50)
```

ตัวอย่าง โปรแกรม 8.3 ใช้สำหรับพิมพ์ภาพจากจอภาพ ซึ่งเป็นภาพที่ได้จากไฟล์ชนิด .GTP ในตัวอย่างนี้ จะพิมพ์ภาพนกซึ่งได้จากไฟล์ BIRD.GTP โปรแกรมมี 2 ส่วน คือ ส่วนแสดงภาพจากไฟล์ และส่วนพิมพ์ภาพจากจอภาพ โปรแกรมจะเริ่มโดยการอ่านข้อมูลจากไฟล์มาแสดงที่จอภาพ และรอการกดคีย์ ถ้าคีย์ P ถูกกด เครื่องพิมพ์จะพิมพ์ภาพ แต่ถ้ากดคีย์อื่นโปรแกรมจะหยุดทำงาน

โปรแกรม 8.3

```
PROGRAM PrintFullScreen;
{print full screen picture}
USES Crt,Graph,Printer;
VAR Ch : Char;

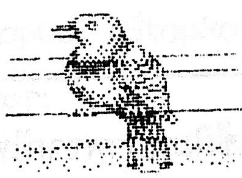
PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;
```

```
BEGIN
  Assign(PictureFile,Filename);
  Reset(PictureFile);
  GetMem(PicturePointer,FileSize(PictureFile));
  BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
  Close(PictureFile);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyFullScreen;
VAR Row,Col,BandNo    : Integer;
    PicCode,N1,N2     : Byte;
BEGIN
  N2 := GetMaxX DIV 256;
  N1 := GetMaxX MOD 256;
  BandNo := 0;  {top row}
  REPEAT
    Write(Lst,#27#51#23);          {set line spacing 23/216 inch}
    Write(Lst,#27#42#6+Chr(N1)+Chr(N2));  {set 90 dot/inch}
    FOR Col := 0 TO GetMaxX DO         {full X print columns}
    BEGIN
      PicCode := 0;              {blank}
      FOR Row := 0+BandNo TO 7+BandNo DO
      BEGIN
        IF GetPixel(Col,Row) <> GetBkColor THEN
        CASE Row-BandNo OF
          0 : PicCode := PicCode + 128;
          1 : PicCode := PicCode +  64;
```

```
          2 : PicCode := PicCode + 32;
          3 : PicCode := PicCode + 16;
          4 : PicCode := PicCode +  8;
          5 : PicCode := PicCode +  4;
          6 : PicCode := PicCode +  2;
          7 : PicCode := PicCode +  1;
        END;
      END;
      Write(Lst,Chr(PicCode));  {print 1 column}
    END;
    Write(Lst,#10);              {line feed}
    Inc(BandNo,8);
  UNTIL BandNo > GetMaxY;        {full row}
  Write(Lst,#12#50);             {form feed & normal spacing}
END;

BEGIN
  PictureFromFile('BIRD.GTP');
  Ch := ReadKey;
  IF Ch IN['P','p'] THEN CopyFullScreen;
  CloseGraph;
END.
```



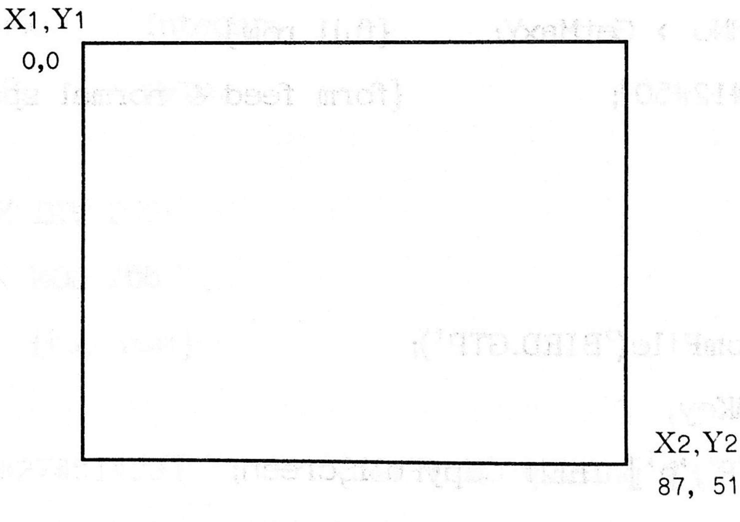
รูปที่ 8.9 ภาพจากโปรแกรม 8.3

8.5 การพิมพ์ภาพบางส่วนจากจอภาพ

การพิมพ์ภาพตามตัวอย่างที่ผ่านมาเป็นการพิมพ์ภาพครั้งละ 1 จอภาพ ไม่ว่าจะมีภาพอยู่เต็มจอภาพหรือไม่ก็ตาม อย่างไรก็ดี เรายังมีวิธีการพิมพ์ภาพเฉพาะบางส่วนของภาพที่ปรากฏบนจอภาพได้อีกด้วย ซึ่งจะช่วยให้ได้ภาพตรงตามความต้องการและใช้เวลาน้อยกว่าการพิมพ์ภาพหมดทั้งจอภาพ

จากวิธีการพิมพ์ภาพหมดทั้งจอภาพ เราสามารถนำมาปรับปรุงให้เป็นการพิมพ์ภาพบางส่วนได้ ซึ่งมีรายการปรับปรุงต่างๆ ดังต่อไปนี้

1. กำหนดขอบเขตของภาพที่ต้องการพิมพ์ โดยกำหนดค่า X1,Y1 ที่ตำแหน่งมุมบนซ้าย และค่า X2, Y2 ที่ตำแหน่งมุมล่างขวาของภาพ ดังตัวอย่างในรูปที่ 8.10



รูปที่ 8.10 ค่า X1,Y1 และ X2,Y2 ของภาพขนาด 88x52 จุด

2. กำหนดให้โปรแกรมอ่าน และพิมพ์ข้อมูลจากจอภาพเฉพาะในขอบเขตที่กำหนดไว้

ตัวอย่าง จากโปรแกรม 8.3 ซึ่งใช้พิมพ์ภาพจากไฟล์ BIRD.GTP จะเห็นได้ว่าภาพมีขนาดเล็กกว่าจอภาพมาก และเครื่องพิมพ์ต้องทำงานนานเพราะต้องเลื่อนหัวพิมพ์ไปมาในส่วนที่ไม่มีภาพด้วย เมื่อปรับปรุงโปรแกรมให้พิมพ์เฉพาะส่วนที่

เป็นภาพ จะได้ดังแสดงในโปรแกรม 8.4 ซึ่งเราสามารถเปรียบเทียบได้ว่าโปรแกรม 8.4 ใช้เวลาพิมพ์น้อยกว่า

โปรแกรม 8.4

```
PROGRAM PrintBlockScreen;
{print a block of screen picture}
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;
BEGIN
  Assign(PictureFile,Filename);
  Reset(PictureFile);
  GetMem(PicturePointer,FileSize(PictureFile));
  BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
  Close(PictureFile);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreen(StartCol,StartRow,
                          StopCol ,StopRow : Integer);
VAR Row,Col,BandNo     : Integer;
    PicCode            : Byte;
BEGIN
  BandNo := StartRow;  {top row}
```

```
  REPEAT
     Write(Lst,#27#51#23);    {set line spacing 23/216 inch}
     FOR Col := StartCol TO StopCol DO
     BEGIN
        PicCode := 0;                  {blank}
        Write(Lst,#27#42#6#1#0);  {set 90 dot/inch, one column}
        FOR Row := 0+BandNo TO 7+BandNo DO
        BEGIN
           IF GetPixel(Col,Row) <> 0 THEN
           CASE Row-BandNo OF
              0 : PicCode := PicCode + 128;
              1 : PicCode := PicCode +  64;
              2 : PicCode := PicCode +  32;
              3 : PicCode := PicCode +  16;
              4 : PicCode := PicCode +   8;
              5 : PicCode := PicCode +   4;
              6 : PicCode := PicCode +   2;
              7 : PicCode := PicCode +   1;
           END;
        END;
        Write(Lst,Chr(PicCode));  {print 1 column}
     END;
     Write(Lst,#10);               {line feed}
     Inc(BandNo,8);
  UNTIL BandNo > StopRow;
  Write(Lst,#12#50);   {form feed & normal spacing}
END;


BEGIN
  PictureFromFile('BIRD.GTP');
```

```
  Ch := ReadKey;
  IF Ch IN['P','p'] THEN CopyBlockScreen(0,0,87,51);
  CloseGraph;
END.
```

8.6 การกำหนดตำแหน่งภาพบนกระดาษ

ตามปกติเครื่องพิมพ์จะเริ่มพิมพ์ที่แถวแรกและคอลัมน์แรกเสมอ ซึ่งเป็นตำแหน่งซ้ายสุดและบนสุดของกระดาษ ถ้าต้องเปลี่ยนตำแหน่งภาพให้เหมาะกับหน้ากระดาษ ก็สามารถทำได้ โดยการกำหนดตำแหน่งบรรทัดและคอลัมน์เริ่มต้น ซึ่งเป็นตำแหน่งมุมบนซ้ายของภาพ ด้วยคำสั่งกำหนดตำแหน่งคอลัมน์เริ่มต้น ซึ่งมีรูปแบบของคำสั่งดังนี้

```
ESC I n
```

โดย n คือ จำนวนคอลัมน์ ซึ่ง 1 คอลัมน์มีขนาดเท่ากับความกว้างของ 1 ตัวอักษร

ตัวอย่าง การกำหนดคอลัมน์เริ่มต้นที่ตำแหน่ง 20 เขียนเป็นสเตตเมนต์ได้ดังนี้

```
Write (Lst, #27#108#20)
```

หรือ

```
Write (Lst, #$1B#$6C#$14)
```

สำหรับการกำหนดตำแหน่งบรรทัดเริ่มต้น ทำได้โดยวิธีการเลื่อนกระดาษขึ้นครั้งละ 1 บรรทัด จนกระทั่งถึงตำแหน่งที่ต้องการ วิธีเลื่อนกระดาษขึ้นหลายบรรทัดอาจใช้ลูป FOR เช่น การเลื่อนกระดาษขึ้น 10 บรรทัด เขียนเป็นสเตตเมนต์ได้ดังนี้

```
FOR Count := 1 TO 10 DO WriteLn (Lst)
```

ตัวอย่าง โปรแกรม 8.5 ใช้สำหรับพิมพ์ภาพที่แสดงบนจอภาพ ซึ่งเป็นภาพที่ได้จากไฟล์ PUPPY.GTP ในโปรแกรมนี้ เราจะต้องกำหนดขอบเขตของภาพเฉพาะส่วนที่

ต้องการพิมพ์จากจอภาพ พร้อมทั้งต้องกำหนดตำแหน่งบรรทัดและคอลัมน์บนกระดาษ ซึ่งเป็นตำแหน่งสำหรับพิมพ์มุมซ้ายบนของภาพ

โปรแกรม 8.5

```
PROGRAM PrintBlockToFrame;
{print a block of screen at the frame}
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;
BEGIN
   Assign(PictureFile,Filename);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockToFrame(StartCol,StartRow,
                           StopCol ,StopRow,
                           Topmargin,LeftMargin : Integer);
VAR Row,Col,BandNo     : Integer;
    PicCode            : Byte;
```

```
BEGIN
  Write(Lst,#27#51#23);          {set line spacing 23/216 inch}
  FOR BandNo := 1 TO TopMargin DO WriteLn(Lst); {set top margin}
  Write(Lst,#27#108+Chr(LeftMargin));              {set left margin}
  BandNo := StartRow;  {top row}
  REPEAT
    Write(Lst,#27#51#23);          {set line spacing 23/216 inch}
    FOR Col := StartCol TO StopCol DO
    BEGIN
      PicCode := 0;                 {blank}
      Write(Lst,#27#42#6#1#0);       {set 90 dot/inch}
      FOR Row := 0+BandNo TO 7+BandNo DO
      BEGIN
        IF GetPixel(Col,Row) <> GetBkColor THEN
        CASE Row-BandNo OF
            0 : PicCode := PicCode + 128;
            1 : PicCode := PicCode +  64;
            2 : PicCode := PicCode +  32;
            3 : PicCode := PicCode +  16;
            4 : PicCode := PicCode +   8;
            5 : PicCode := PicCode +   4;
            6 : PicCode := PicCode +   2;
            7 : PicCode := PicCode +   1;
        END;
      END;
      Write(Lst,Chr(PicCode));   {print 1 column}
    END;
    Write(Lst,#10);                   {line feed}
    Inc(BandNo,8);
```

```
   UNTIL BandNo > StopRow;
   Write(Lst,#12#50); {formfeed & normal spacing}
END;

BEGIN
   PictureFromFile('PUPPY.GTP');
   Ch := ReadKey;
   IF Ch IN['P','p'] THEN CopyBlockToFrame(0,0,87,51,10,20);
   CloseGraph;
END.
```

8.7 การพิมพ์ภาพแบบต่างๆ

การพิมพ์ภาพจากจอภาพ นอกจากจะพิมพ์ให้เหมือนกับภาพที่เห็นบนจอภาพแล้ว เรายังสามารถพิมพ์ภาพให้แตกต่างไปจากเดิมได้อีกหลายแบบ เช่น แบบกลับสี กลับบน-ล่าง กลับซ้าย-ขวา แบบขยายและแบบย่อ ซึ่งแต่ละแบบมีรายละเอียดดังต่อไปนี้

8.7.1 การพิมพ์ภาพแบบกลับสี หมายถึงการพิมพ์ภาพให้มีสีตรงกันข้ามกับที่มองเห็นบนจอภาพ ซึ่งจากวิธีการพิมพ์ภาพที่ผ่านมาเรามีหลักว่าตำแหน่งที่เป็นจุดสว่างบนจอภาพจะเป็นจุดดำบนกระดาษ ถ้าใช้หลักการกลับกันคือ ตำแหน่งที่เป็นจุดสว่างบนจอภาพจะเป็นจุดขาวบนกระดาษ และตำแหน่งที่เป็นจุดมืด (จุดที่มีสีเหมือนสีพื้น) บนจอภาพจะเป็นจุดดำบนกระดาษ ภาพที่พิมพ์ได้จะมีสีกลับกันกับภาพที่พิมพ์ด้วยวิธีแรกแบบจุดต่อจุด นั่นคือจุดดำในภาพแรกจะเป็นจุดขาวในภาพหลัง และจุดขาวในภาพแรกจะเป็นจุดดำในภาพหลัง

การพิมพ์ภาพแบบกลับสี ทำได้โดยการเปลี่ยนข้อมูลที่จะส่งไปยังเครื่องพิมพ์ให้เป็นค่าตรงกันข้าม เพื่อให้เครื่องพิมพ์ใช้เข็มในตำแหน่งตรงข้ามกัน การเปลี่ยนข้อมูลภาพให้เป็นค่าตรงกันข้ามกับค่าเดิม สามารถทำได้ง่ายๆ โดยการนำค่าเดิมที่มีอยู่ไปลบออกจาก 255 ซึ่งเขียนเป็นสมการได้ดังนี้

ข้อมูลใหม่ = 255 – ข้อมูลเดิม

ตัวอย่างข้อมูลซึ่งเป็นค่าตรงข้ามกันและตำแหน่งของเข็มที่ใช้สำหรับพิมพ์ข้อมูลนั้นแสดงในตารางที่ 8.2

ตารางที่ 8.2 ตัวอย่างข้อมูลภาพ 2 ชุดซึ่งมีค่าตรงข้ามกัน

ข้อมูลเดิม	เข็มที่พิมพ์	ข้อมูลใหม่	เข็มที่พิมพ์
0	ไม่มี	255	0–7
1	0	254	1–7
2	1	253	0,2–7
3	0–1	252	2–7
24	3–4	231	0–2,5–7
129	0,7	126	1–6
255	0–7	0	ไม่มี

ตัวอย่าง โปรแกรม 8.6 ใช้สำหรับพิมพ์ภาพที่มีสีกลับกันกับภาพที่อยู่บนจอภาพ ซึ่งในตัวอย่างนี้เป็นภาพที่ได้จากไฟล์ LADY.GTP ขอให้สังเกตข้อมูลที่โปรแกรมส่งไปพิมพ์ และเปรียบเทียบภาพที่พิมพ์แบบปกติกับพิมพ์แบบกลับสี

โปรแกรม 8.6

```
PROGRAM PrintBlockScreenReverse;
{print reverse picture}
USES Crt,Graph,Printer;
VAR Ch : Char;
```

```
PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode  : Integer;
     PicturePointer         : Pointer;
     PictureFile            : File;
BEGIN
   Assign(PictureFile,Filename);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreen(StartCol,StartRow,
                          StopCol ,StopRow : Integer);
VAR Row,Col,BandNo    : Integer;
    PicCode           : Byte;
BEGIN
   BandNo := StartRow;  {top row}
   REPEAT
      Write(Lst,#27#51#23);    {set line spacing 23/216 inch}
      FOR Col := StartCol TO StopCol DO
      BEGIN
         PicCode := 0;               {blank}
         Write(Lst,#27#42#6#1#0);  {set 90 dot/inch, one column}
         FOR Row := 0+BandNo TO 7+BandNo DO
         BEGIN
            IF GetPixel(Col,Row) <> GetBkColor THEN
```

```
            CASE Row-BandNo OF
                0 : PicCode := PicCode + 128;
                1 : PicCode := PicCode +  64;
                2 : PicCode := PicCode +  32;
                3 : PicCode := PicCode +  16;
                4 : PicCode := PicCode +   8;
                5 : PicCode := PicCode +   4;
                6 : PicCode := PicCode +   2;
                7 : PicCode := PicCode +   1;
            END;
         END;
         PicCode := 255 - PicCode; {reverse picture}
         Write(Lst,Chr(PicCode));   {print 1 column}
      END;
      Write(Lst,#10);                {line feed}
      Inc(BandNo,8);
   UNTIL BandNo > StopRow;
   Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
   PictureFromFile('LADY.GTP');
   Ch := ReadKey;
   IF Ch IN['P','p'] THEN CopyBlockScreen(8,0,519,284);
   CloseGraph;
END.
```





รูปที่ 8.11 ภาพจากโปรแกรม 8.6

ตัวอย่าง โปรแกรม 8.7 ใช้สำหรับพิมพ์ภาพกลับสีกับจอภาพ ในโปรแกรมนี้ใช้ภาพจากไฟล์ BIRD.GTP ขอให้สังเกต การกำหนดขอบเขตของภาพที่จะพิมพ์ ข้อมูลที่ส่งพิมพ์ และเปรียบเทียบกับภาพที่พิมพ์แบบปกติ

โปรแกรม 8.7

```
PROGRAM PrintBlockScreenReverse;
{print reverse picture}
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;
BEGIN
  Assign(PictureFile,Filename);
  Reset(PictureFile);
  GetMem(PicturePointer,FileSize(PictureFile));
  BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
  Close(PictureFile);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreen(StartCol,StartRow,
                          StopCol ,StopRow : Integer);
VAR Row,Col,BandNo    : Integer;
```

โปรแกรม 8.8

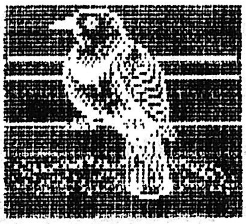
```
PROGRAM PrintBlockScreenUpEnd;
{print up end block from screen }
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode    : Integer;
     PicturePointer           : Pointer;
     PictureFile              : File;
BEGIN
   Assign(PictureFile,Filename);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreenUpEnd(StartCol,StartRow,
                               StopCol ,StopRow : Integer);
VAR Row,Col,BandNo     : Integer;
    PicCode            : Byte;
BEGIN
   BandNo := StopRow;  {bottom row}
   REPEAT
      Write(Lst,#27#51#23);    {set line spacing 23/216 inch}
```

```
  Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
  PictureFromFile('BIRD.GTP');
  Ch := ReadKey;
  IF Ch IN['P','p'] THEN CopyBlockScreen(0,0,87,51);
  CloseGraph;
END.
```



รูปที่ 8.12 ภาพจากโปรแกรม 8.7

8.7.2 การพิมพ์ภาพแบบกลับบน-ล่าง หมายถึง การพิมพ์ส่วนที่อยู่ด้านบนของภาพบนจอภาพให้เป็นส่วนล่างของภาพบนกระดาษ ภาพที่ได้จะเป็นภาพกลับหัว ซึ่งถ้าบนจอภาพเป็นภาพคนยืนที่พื้น ภาพบนกระดาษจะเป็นภาพคนยืนที่เพดาน โดยศีรษะอยู่ข้างล่างและเท้าอยู่ข้างบน

การพิมพ์ภาพแบบนี้ ทำได้โดยการอ่านข้อมูลภาพบนจอภาพจากด้านล่างขึ้นข้างบน แล้วส่งให้เครื่องพิมพ์ทำการพิมพ์ตามลำดับ เนื่องจากเครื่องพิมพ์ต้องพิมพ์จากข้างบนลงข้างล่าง ดังนั้นภาพที่ได้จึงมีลักษณะกลับบน–ล่างกับภาพที่เห็นบนจอภาพ

ตัวอย่าง โปรแกรม 8.8 ใช้สำหรับพิมพ์ภาพแบบกลับบน–ล่างกับภาพบนจอภาพ โดยใช้ภาพซึ่งได้จากไฟล์ PUPPY.GTP ขอให้สังเกตการอ่านข้อมูลบนจอภาพของโปรแกรม ซึ่งจะอ่านจากขอบเขตที่กำหนดไว้บนจอภาพ

```
   PicCode          : Byte;
BEGIN
   BandNo := StartRow;   {top row}
   REPEAT
      Write(Lst,#27#51#23);    {set line spacing 23/216 inch}
      FOR Col := StartCol TO StopCol DO
      BEGIN
         PicCode := 0;                  {blank}
         Write(Lst,#27#42#6#1#0);   {set 90 dot/inch, one column}
         FOR Row := 0+BandNo TO 7+BandNo DO
         BEGIN
            IF GetPixel(Col,Row) <> GetBkColor THEN
            CASE Row-BandNo OF
               0 : PicCode := PicCode + 128;
               1 : PicCode := PicCode +  64;
               2 : PicCode := PicCode +  32;
               3 : PicCode := PicCode +  16;
               4 : PicCode := PicCode +   8;
               5 : PicCode := PicCode +   4;
               6 : PicCode := PicCode +   2;
               7 : PicCode := PicCode +   1;
            END;
         END;
         PicCode := 255 - PicCode; {reverse picture}
         Write(Lst,Chr(PicCode));   {print 1 column}
      END;
      Write(Lst,#10);                  {line feed}
      Inc(BandNo,8);
   UNTIL BandNo > StopRow;
```

```
    FOR Col := StartCol TO StopCol DO
    BEGIN
      PicCode := 0;                  {blank}
      Write(Lst,#27#42#6#1#0);  {set 90 dot/inch, one column}
      FOR Row := 0+BandNo DOWNTO BandNo-7 DO
      BEGIN
        IF GetPixel(Col,Row) <> GetBkColor THEN
        CASE BandNo-Row OF    { negative }
          0 : PicCode := PicCode + 128;
          1 : PicCode := PicCode +  64;
          2 : PicCode := PicCode +  32;
          3 : PicCode := PicCode +  16;
          4 : PicCode := PicCode +   8;
          5 : PicCode := PicCode +   4;
          6 : PicCode := PicCode +   2;
          7 : PicCode := PicCode +   1;
        END;
      END;
      Write(Lst,Chr(PicCode));  {print 1 column}
    END;
    Write(Lst,#10);                {line feed}
    Dec(BandNo,8);                 {upper row}
  UNTIL BandNo < StartRow;
  Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
  PictureFromFile('PUPPY.GTP');
  Ch := ReadKey;
```

```
  IF Ch IN['P','p'] THEN CopyBlockScreenUpEnd(0,0,87,51);
  CloseGraph;
END.
```



รูปที่ 8.13 ภาพจากโปรแกรม 8.8

8.7.3 การพิมพ์ภาพแบบกลับซ้าย-ขวา หมายถึง การพิมพ์ส่วนที่อยู่ด้านซ้ายของภาพบนจอภาพให้เป็นส่วนขวาของภาพบนกระดาษ ภาพที่ได้จะคล้ายกับภาพที่เรามองเห็นในกระจกเงา

การพิมพ์ภาพแบบนี้ ทำได้โดยการอ่านข้อมูลภาพบนจอภาพจากทางด้านขวามือมาทางด้านซ้ายมือ แล้วส่งให้เครื่องพิมพ์ทำการพิมพ์ตามลำดับ เนื่องจากเครื่องพิมพ์ต้องพิมพ์จากซ้ายไปขวา ดังนั้นภาพที่ได้จึงมีลักษณะกลับซ้าย-ขวากับภาพบนจอภาพ

ตัวอย่าง โปรแกรม 8.9 ใช้สำหรับพิมพ์ภาพแบบกลับซ้าย-ขวากับภาพบนจอภาพ โดยใช้ภาพซึ่งได้จากไฟล์ PUPPY.GTP ขอให้สังเกตการอ่านข้อมูลบนจอภาพของโปรแกรม ซึ่งจะอ่านจากขอบเขตที่กำหนดไว้บนจอภาพ

โปรแกรม 8.9

```
PROGRAM PrintBlockScreenReflect;
{print reflect block from screen }
USES Crt,Graph,Printer;
VAR Ch : Char;
```

```
PROCEDURE  PictureFromFile(Filename : String);
VAR GraphDriver,GraphMode  : Integer;
    PicturePointer         : Pointer;
    PictureFile            : File;
BEGIN
  Assign(PictureFile,Filename);
  Reset(PictureFile);
  GetMem(PicturePointer,FileSize(PictureFile));
  BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
  Close(PictureFile);
  GraphDriver := Detect;
  Initgraph(GraphDriver,GraphMode,'');
  PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreenReflect(StartCol,StartRow,
                                 StopCol ,StopRow : Integer);
VAR Row,Col,BandNo     : Integer;
    PicCode            : Byte;
BEGIN
  BandNo := StartRow;  {top row}
  REPEAT
    Write(Lst,#27#51#23);    {set line spacing 23/216 inch}
    FOR Col := StopCol DOWNTO StartCol DO
    BEGIN
      PicCode := 0;                 {blank}
      Write(Lst,#27#42#6#1#0);  {set 90 dot/inch, one column}
      FOR Row := 0+BandNo TO 7+BandNo DO
      BEGIN
        IF GetPixel(Col,Row) <> GetBkColor THEN
```

```
          CASE Row-BandNo OF
            0 : PicCode := PicCode + 128;
            1 : PicCode := PicCode +  64;
            2 : PicCode := PicCode +  32;
            3 : PicCode := PicCode +  16;
            4 : PicCode := PicCode +   8;
            5 : PicCode := PicCode +   4;
            6 : PicCode := PicCode +   2;
            7 : PicCode := PicCode +   1;
          END;
        END;
        Write(Lst,Chr(PicCode));  {print 1 column}
      END;
      Write(Lst,#10);                 {line feed}
      Inc(BandNo,8);
    UNTIL BandNo > StopRow;
    Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
   PictureFromFile('PUPPY.GTP');
   Ch := ReadKey;
   IF Ch IN['P','p'] THEN CopyBlockScreenReflect(0,0,87,51);
   CloseGraph;
END.
```



รูปที่ 8.14 ภาพจากโปรแกรม 8.9

จากโปรแกรม 8.9 เราสามารถเปลี่ยนแปลงที่โปรแกรมหลักเพื่อใช้พิมพ์ภาพแบบกลับซ้าย-ขวากับภาพบนจอภาพซึ่งได้จากไฟล์ LADY.GTP สเตตเมนต์ในโปรแกรมหลักเป็นดังนี้

```
BEGIN
    PictureFromFile('LADY.GTP');
    Ch := ReadKey;
    IF Ch IN['P','p'] THEN CopyBlockScreenReflect
                                   (8,0,519,284);
    CloseGraph
END.
```

8.7.4 การพิมพ์ภาพแบบขยาย หมายถึง การพิมพ์ภาพให้มีขนาดโตกว่าที่ได้จากการพิมพ์ตามปกติ การพิมพ์ภาพขยายสามารถทำได้ทั้งการขยายในทางสูง และการขยายในทางกว้าง ซึ่งมีรายละเอียดดังต่อไปนี้

1. การขยายภาพในทางกว้าง หมายถึง การพิมพ์ภาพให้มีขนาดกว้างมากขึ้นแต่ส่วนสูงของภาพไม่เปลี่ยนแปลง วิธีพิมพ์ภาพแบบนี้มี 2 วิธีคือ

ก. โดยการกำหนดค่า m ในคำสั่งกำหนดโหมดพิมพ์กราฟิกส์ ซึ่งได้กล่าวมาแล้วในข้อ 8.3.1 คือ คำสั่ง

ESC * m n1 n2 ข้อมูล

จากตารางที่ 8.1 จะเห็นได้ว่าถ้า m = 0 ระยะห่างระหว่างแต่ละจุดจะมีค่าสูงสุด ซึ่งหมายความว่าภาพจะมีความกว้างมากที่สุด ถ้า m=3 ระยะห่างระหว่างแต่ละจุดจะมีค่าต่ำสุด ภาพที่พิมพ์ได้จะมีความกว้างน้อยที่สุด ดังนั้น ค่า m เรียงตามลำดับจากภาพกว้างที่สุดไปยังภาพกว้างน้อยที่สุดจึงเท่ากับ 0 5 4 6 1 2 และ 3 ทั้งนี้มีข้อสังเกตที่น่าสนใจคือ

(1) เครื่องพิมพ์กำหนดให้ m = 0 เป็นระยะห่างมาตรฐาน

(2) ค่า m = 1 และ m = 2 พิมพ์ภาพได้กว้างเท่ากันแต่ m = 2 พิมพ์ได้เร็วกว่า

(3) ขนาดภาพที่มีสัดส่วนคล้ายกับภาพที่ปรากฏบนจอภาพ จะได้จากการกำหนดค่า m = 6 และ 1 หรือ 2

ข. ขยายความกว้างของภาพโดยการอ่านจุดจากจอภาพ 1 คอลัมน์ แต่นำมาพิมพ์หลายคอลัมน์ เช่น อ่านภาพ 1 คอลัมน์ พิมพ์ 2 คอลัมน์ จะได้ภาพกว้าง 2 เท่า
อ่านภาพ 1 คอลัมน์ พิมพ์ 3 คอลัมน์ จะได้ภาพกว้าง 3 เท่า
อ่านภาพ 1 คอลัมน์ พิมพ์ 4 คอลัมน์ จะได้ภาพกว้าง 4 เท่า

จะเห็นได้ว่า การขยายภาพด้วยวิธีการนี้ เราสามารถระบุจำนวนเท่าของการขยายได้ชัดเจนกว่า สเตตเมนต์สำหรับพิมพ์หลายคอลัมน์ซึ่งอยู่ในโพรซีเยอร์ Copy-BlockScreenW เป็นดังนี้

```
FOR Count := 1 TO WidthScale DO Write(Lst,Chr(PicCode))
```

โดยมีการเพิ่มพารามิเตอร์สำหรับกำหนดจำนวนเท่า ของการขยายภาพขึ้นอีก 1 ตัวคือ WidthScale

---

ตัวอย่าง โปรแกรม 8.10 ใช้สำหรับพิมพ์ภาพขยายในทางกว้าง โดยใช้ภาพบนจอภาพซึ่งได้จากไฟล์ BIRD.GTP โปรแกรมจะพิมพ์ภาพตามขนาดของค่า m ที่กำหนดให้ ซึ่งเป็นได้ทั้งภาพขยายและภาพย่อ ในที่นี้กำหนดให้ m = 0 ภาพที่ได้จะเป็นภาพขยายกว้างกว่าปกติ

โปรแกรม 8.10

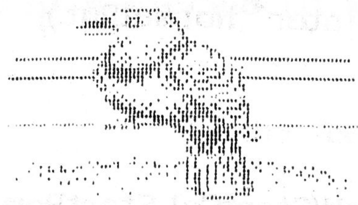
```
PROGRAM PrintBlockScreenMode;
{print a block of screen with vary print mode}
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;
BEGIN
  Assign(PictureFile,Filename);
  Reset(PictureFile);
  GetMem(PicturePointer,FileSize(PictureFile));
  BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
  Close(PictureFile);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreenM(StartCol,StartRow,
                           StopCol,StopRow,PrintMode : Integer);
VAR Row,Col,BandNo     : Integer;
   PicCode             : Byte;
BEGIN
  BandNo := StartRow;  {top row}
  REPEAT
    Write(Lst,#27#51#23);    {set line spacing 23/216 inch}
    FOR Col := StartCol TO StopCol DO
```

```
      BEGIN
        PicCode := 0;                 {blank}
        Write(Lst,#27#42+Chr(PrintMode)+#1#0); {vary print mode}
        FOR Row := 0+BandNo TO 7+BandNo DO
        BEGIN
          IF GetPixel(Col,Row) <> 0 THEN
          CASE Row-BandNo OF
            0 : PicCode := PicCode + 128;
            1 : PicCode := PicCode + 64;
            2 : PicCode := PicCode + 32;
            3 : PicCode := PicCode + 16;
            4 : PicCode := PicCode +  8;
            5 : PicCode := PicCode +  4;
            6 : PicCode := PicCode +  2;
            7 : PicCode := PicCode +  1;
          END;
        END;
        Write(Lst,Chr(PicCode));   {print 1 column}
      END;
      Write(Lst,#10);                 {line feed}
      Inc(BandNo,8);
   UNTIL BandNo > StopRow;
   Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
   PictureFromFile('BIRD.GTP');
   Ch := ReadKey;
   IF Ch IN['P','p'] THEN CopyBlockScreenM(0,0,87,51,0);
   CloseGraph;
END.
```



รูปที่ 8.15 ภาพจากโปรแกรม 8.10

ตัวอย่าง โปรแกรม 8.11 ใช้สำหรับพิมพ์ภาพขยายในทางกว้าง โดยใช้ภาพบนจอภาพซึ่งได้จากไฟล์ PUPPY.GTP โปรแกรมจะพิมพ์ภาพกว้างออกตามขนาดของค่าพารามิเตอร์ WidthScale ซึ่งค่าดังกล่าวนี้จะต้องเป็นค่าที่มากกว่า 1

โปรแกรม 8.11

```
PROGRAM PrintWidePictureFromScreen;
{print wide picture}
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode   : Integer;
     PicturePointer          : Pointer;
     PictureFile             : File;
BEGIN
  Assign(PictureFile,Filename);
  Reset(PictureFile);
  GetMem(PicturePointer,FileSize(PictureFile));
  BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
  Close(PictureFile);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
```

```
   PutImage(0,0,PicturePointer^,NormalPut);
END;


PROCEDURE CopyBlockScreenW(StartCol,StartRow,StopCol ,StopRow,
                            WidthScale : Integer);
VAR Row,Col,BandNo,Count   : Integer;
    PicCode              : Byte;
BEGIN
   BandNo := StartRow;  {top row}
   REPEAT
      Write(Lst,#27#51#23);    {set line spacing 23/216 inch}
      FOR Col := StartCol TO StopCol DO
      BEGIN
         PicCode := 0;                {blank}
         Write(Lst,#27#42#6+Chr(WidthScale)+#0); {set 90 dot/inch}
         FOR Row := 0+BandNo TO 7+BandNo DO
         BEGIN
            IF GetPixel(Col,Row) <> GetBkColor THEN
            CASE Row-BandNo OF
               0 : PicCode := PicCode + 128;
               1 : PicCode := PicCode +  64;
               2 : PicCode := PicCode +  32;
               3 : PicCode := PicCode +  16;
               4 : PicCode := PicCode +   8;
               5 : PicCode := PicCode +   4;
               6 : PicCode := PicCode +   2;
               7 : PicCode := PicCode +   1;
            END;
         END;
```

```
        FOR Count := 1 TO WidthScale DO Write(Lst,Chr(PicCode));
      END;
      Write(Lst,#10);                 {line feed}
      Inc(BandNo,8);
    UNTIL BandNo > StopRow;
    Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
  PictureFromFile('PUPPY.GTP');
  Ch := ReadKey;
  IF Ch IN['P','p'] THEN CopyBlockScreenW(0,0,87,51,2);
  CloseGraph;
END.
```



รูปที่ 8.16 ภาพจากโปรแกรม 8.11

2. การขยายภาพในทางสูงหมายถึงการพิมพ์ภาพให้มีความสูงมากขึ้น แต่ความกว้างของภาพไม่เปลี่ยนแปลงวิธีพิมพ์ภาพแบบนี้ทำได้โดยการส่งข้อมูลจากจอภาพ 1 จุด ให้แก่เข็มของเครื่องพิมพ์มากกว่า 1 เข็ม เช่น ถ้าส่งข้อมูล 1 จุด ให้แก่เข็ม 2 เข็ม ภาพจะสูงกว่าเดิม 2 เท่า ถ้าส่งข้อมูล 1 จุดให้แก่เข็ม 3 เข็ม ภาพจะสูงกว่าเดิม 3 เท่า เป็นเช่นนี้ไปตามลำดับ

เนื่องจากเครื่องพิมพ์ใช้เข็ม 8 เข็มสำหรับการพิมพ์ 1 คอลัมน์ ตามปกติแต่ละเข็มจะพิมพ์ข้อมูลจากจอภาพ 1 จุด ถ้ามีการส่งข้อมูลจากจอภาพ 1 จุดให้แก่เข็มถึง 2 เข็ม จำนวนจุดที่ส่งได้แต่ละครั้งจะเหลือเพียง 4 จุด เพราะฉะนั้นการอ่านข้อมูลจากจอภาพจึงต้องอ่านแถบละ 4 จุด เวลาที่ใช้ในการพิมพ์จะเพิ่มขึ้นประมาณ 2 เท่าด้วย

ตัวอย่าง โปรแกรม 8.12 ใช้สำหรับพิมพ์ภาพขยายในทางสูง 2 เท่า โดยใช้ภาพบนจอภาพซึ่งได้จากไฟล์ PUPPY.GTP ขอให้สังเกตการอ่านข้อมูลภาพครั้งละ 4 จุด และการส่งข้อมูลให้แก่เข็มครั้งละ 2 เข็ม

โปรแกรม 8.12

```
PROGRAM PrintHigh2;
{print 2 time hight picture on sreen}
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode  : Integer;
     PicturePointer         : Pointer;
     PictureFile            : File;
BEGIN
   Assign(PictureFile,Filename);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
```

```
  PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreenH2(StartCol,StartRow,
                            StopCol ,StopRow : Integer);
VAR Row,Col,BandNo    : Integer;
    PicCode           : Byte;
BEGIN
  BandNo := StartRow;  {top row}
  REPEAT
    Write(Lst,#27#51#23);    {set line spacing 23/216 inch}
    FOR Col := StartCol TO StopCol DO
    BEGIN
      PicCode := 0;                {blank}
      Write(Lst,#27#42#6#1#0);  {set 90 dot/inch, one column}
      FOR Row := 0+BandNo TO 3+BandNo DO  {read 4 row}
      BEGIN
        IF GetPixel(Col,Row) <> GetBkColor THEN
        CASE Row-BandNo OF
          0 : PicCode := PicCode + 128 + 64;
          1 : PicCode := PicCode +  32 + 16;
          2 : PicCode := PicCode +   8 +  4;
          3 : PicCode := PicCode +   2 +  1;
        END;
      END;
      Write(Lst,Chr(PicCode));  {print 1 column}
    END;
    Write(Lst,#10);               {line feed}
    Inc(BandNo,4);
```

```
   UNTIL BandNo > StopRow;
   Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
   PictureFromFile('PUPPY.GTP');
   Ch := ReadKey;
   IF Ch IN['P','p'] THEN CopyBlockScreenH2(0,0,87,51);
   CloseGraph;
END.
```



รูปที่ 8.17 ภาพจากโปรแกรม 8.12

เมื่อพิจารณาถึงการส่งข้อมูล 1 จุดจากจอภาพ ให้แก่เข็ม 3 เข็ม จะพบว่าจำนวนจุดสูงสุดที่ส่งได้แต่ละครั้งจะมีเพียง 2 จุด โดยส่งให้แก่เข็ม 6 เข็ม คงเหลือเข็มว่าง 2 เข็ม ซึ่งเป็นเข็มที่อยู่ด้านล่าง บริเวณเข็มว่างจะไม่มีการพิมพ์ใดๆ ถ้าเรายังคงใช้ระยะห่างระหว่างบรรทัดขนาดเดิมคือ 23/216 ภาพจะถูกแยกออกเป็นริ้วๆ โดยมีแถบสีขาว (หรือดำ) สลับกับแถบของภาพ ดังนั้นเพื่อแก้ไขข้อบกพร่องดังกล่าวนี้ เราจึงต้องคำนวณระยะห่างระหว่างบรรทัดใหม่ โดยใช้แนวความคิดที่ได้กล่าวแล้วในข้อ 8.3.3 ดังต่อไปนี้

ระยะบรรทัดที่ต้องการ = ความสูง 6 จุด + ระยะห่าง 1 จุด
= 0.07128 + 0.01188
= 0.08316 นิ้ว

หาค่า n จากสมการต่อไปนี้

ระยะบรรทัดที่ต้องการ = n / 216
0.08316 = n / 216
n = 17.96256

ค่า n ที่เหมาะสมจึงควรเท่ากับ 18

สเตตเมนต์ตั้งระยะห่างระหว่างบรรทัดเป็นดังนี้

```
Write (Lst, #27#51#18)
```

หรือ

```
Write (Lst, #$1B#$33#$12)
```

ตัวอย่าง โปรแกรม 8.13 ใช้สำหรับพิมพ์ภาพขยายในทางสูง 3 เท่า โดยใช้ภาพบนจอภาพซึ่งได้จากไฟล์ PUPPY.GTP ขอให้สังเกตการอ่านข้อมูลภาพครั้งละ 2 จุด การส่งข้อมูลให้แก่เข็มครั้งละ 3 เข็ม และ การจัดระยะห่างระหว่างบรรทัดให้มีค่าน้อยลง

โปรแกรม 8.13

```
PROGRAM PrintScreenHigh3;
{print 3 time hight picture on sreen}
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode  : Integer;
```

```
    PicturePointer          : Pointer;
    PictureFile             : File;
BEGIN
   Assign(PictureFile,Filename);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreenH3(StartCol,StartRow,
                            StopCol ,StopRow : Integer);
VAR Row,Col,BandNo      : Integer;
    PicCode             : Byte;
BEGIN
   BandNo := StartRow;  {top row}
   REPEAT
      Write(Lst,#27#51#18);    {set line spacing 18/216 inch}
      FOR Col := StartCol TO StopCol DO
      BEGIN
         PicCode := 0;                  {blank}
         Write(Lst,#27#42#6#1#0);  {set 90 dot/inch, one column}
         FOR Row := 0+BandNo TO 1+BandNo DO  {read 2 row}
         BEGIN
            IF GetPixel(Col,Row) <> GetBkColor THEN
            CASE Row-BandNo OF
               0 : PicCode := PicCode + 128 + 64 + 32;
```

```
        1 : PicCode := PicCode +  16 +  8 +  4;
      END;
    END;
    Write(Lst,Chr(PicCode));   {print 1 column}
  END;
  Write(Lst,#10);                {line feed}
  Inc(BandNo,2);
UNTIL BandNo > StopRow;
Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
  PictureFromFile('PUPPY.GTP');
  Ch := ReadKey;
  IF Ch IN['P','p'] THEN CopyBlockScreenH3(0,0,87,51);
  CloseGraph;
END.
```



รูปที่ 8.18 ภาพจากโปรแกรม 8.13

จากหลักการและตัวอย่างโปรแกรมที่ผ่านมาทำให้เราได้แนวทางที่จะปรับปรุงไปใช้กับการขยายภาพในทางสูงขนาดอื่นๆ ได้ด้วย

3. การขยายภาพทั้งในทางสูงและทางกว้างหมายถึง การพิมพ์ภาพให้กว้างและสูงกว่าภาพเดิม เพื่อให้ได้ภาพซึ่งมีสัดส่วนตามที่ต้องการ วิธีการพิมพ์ภาพแบบนี้ทำได้โดยการรวมวิธีการขยายภาพทั้งในทางสูงและในทางกว้างเข้าด้วยกัน

ตัวอย่าง โปรแกรม 8.14 ใช้สำหรับพิมพ์ภาพขยายทั้งในทางสูงและในทางกว้าง ซึ่งผู้ใช้โปรแกรมสามารถเลือกกำหนดขนาดความกว้างให้แก่ WidthScale ได้ตามต้องการ แต่การขยายในทางสูงถูกกำหนดเป็นค่าคงที่ไว้เป็น 3 เท่า ในตัวอย่างนี้โปรแกรมจะพิมพ์ภาพจากไฟล์ PUPPY.GTP โดยกำหนดขนาดขยายในทางกว้างเป็น 3 เท่า

โปรแกรม 8.14

```
PROGRAM PrintScreenHigh3WidthX;
{print 3 time hight picture with spacify width}
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode    : Integer;
     PicturePointer           : Pointer;
     PictureFile              : File;
BEGIN
   Assign(PictureFile,Filename);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
```

```
  Close(PictureFile);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreenH3(StartCol,StartRow, StopCol,StopRow,
                            WidthScale : Integer);
VAR Row,Col,BandNo, Count   : Integer;
    PicCode                 : Byte;
BEGIN
  BandNo := StartRow;  {top row}
  REPEAT
    Write(Lst,#27#51#18);     {set line spacing 18/216 inch}
    FOR Col := StartCol TO StopCol DO
    BEGIN
      PicCode := 0;               {blank}
      Write(Lst,#27#42#6+Chr(WidthScale)+#0); {set 90 dot/inch}
      FOR Row := 0+BandNo TO 1+BandNo DO  {read 2 row}
      BEGIN
        IF GetPixel(Col,Row) <> GetBkColor THEN
        CASE Row-BandNo OF
          0 : PicCode := PicCode + 128 + 64 + 32;
          1 : PicCode := PicCode +  16 +  8 +  4;
        END;
      END;
      FOR Count := 1 TO WidthScale DO Write(Lst,Chr(PicCode));
    END;
    Write(Lst,#10);    {line feed}
```

```
    Inc(BandNo,2);
  UNTIL BandNo > StopRow;
  Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
  PictureFromFile('PUPPY.GTP');
  Ch := ReadKey;
  IF Ch IN['P','p'] THEN CopyBlockScreenH3(0,0,87,51,3);
  CloseGraph;
END.
```



รูปที่ 8.19 ภาพจากโปรแกรม 8.14

8.7.5 การพิมพ์ภาพแบบย่อ หมายถึง การพิมพ์ภาพให้มีขนาดเล็กกว่าที่ได้จากการพิมพ์ตามปกติ การพิมพ์ภาพแบบย่อสามารถทำได้ทั้งการย่อในทางสูงและการย่อในทางกว้างซึ่งมีรายละเอียดดังต่อไปนี้

1. การย่อภาพในทางกว้าง หมายถึง การพิมพ์ภาพให้มีขนาดแคบลง แต่ส่วนสูงของภาพไม่เปลี่ยนแปลง วิธีพิมพ์ภาพแบบนี้มี 2 วิธีคือ

ก. โดยการกำหนดค่า m ในคำสั่งกำหนดโหมดพิมพ์กราฟิกส์ ซึ่งได้กล่าวมาแล้วในข้อ 8.3.1 คือ คำสั่ง

ESC * m n1 n2 ข้อมูล

จากตารางที่ 8.1 จะเห็นได้ว่าถ้า m = 3 ระยะห่างระหว่างแต่ละจุดจะมีค่าต่ำสุด ซึ่งหมายความว่าภาพจะมีความกว้างน้อยที่สุด

ตัวอย่าง จากโปรแกรม 8.10 ซึ่งใช้สำหรับพิมพ์ภาพขยายในทางกว้าง โดยใช้ภาพจากไฟล์ BIRD.GTP โปรแกรมจะพิมพ์ภาพตามขนาดของค่า m ที่กำหนดให้ ซึ่งเป็นได้ทั้งภาพขยายและภาพย่อ ในที่นี้เราต้องการพิมพ์ภาพย่อ ซึ่งจะทำได้โดยการเปลี่ยนสเตตเมนต์ต่อไปนี้

```
IF Ch IN['P','p'] THEN CopyBlockScreenM(0,0,87,51,0)
```

ให้เป็นสเตตเมนต์ต่อไปนี้

```
IF Ch IN['P','p'] THEN CopyBlockScreenM(0,0,87,51,3)
```

ข. ย่อภาพโดยวิธีการอ่านภาพจากจอภาพ แล้วนำมาพิมพ์เฉพาะคอลัมน์ที่เป็นเลขคู่ (หรือเลขคี่) ซึ่งจะมีการพิมพ์ 1 คอลัมน์เว้น 1 คอลัมน์ ทำให้ภาพที่ได้มีความกว้างเป็นครึ่งหนึ่งของภาพเดิม แต่ภาพที่ได้จากการย่อด้วยวิธีนี้ จะขาดรายละเอียดไปบางส่วน เนื่องจากมีจุดหายไปในคอลัมน์ที่ไม่ได้พิมพ์ เพราะฉะนั้นการย่อภาพในทางกว้างจึงน่าจะใช้วิธีการตามข้อ ก.

2. การย่อภาพในทางสูง หมายถึงการพิมพ์ภาพให้มีขนาดต่ำลง แต่ส่วนกว้างของภาพไม่เปลี่ยนแปลง ซึ่งสามารถทำได้โดยการอ่านภาพจากจอภาพ แล้วนำมาพิมพ์เฉพาะแถวที่เป็นเลขคู่ (หรือเลขคี่) ซึ่งจะมีการพิมพ์ 1 แถวเว้น 1 แถว ทำให้ภาพที่ได้มีความสูงเป็นครึ่งหนึ่งของภาพเดิม แต่ภาพที่ได้จากการย่อด้วยวิธีนี้ จะขาดรายละเอียดไปบางส่วน เนื่องจากมีจุดหายไปในแถวที่ไม่ได้พิมพ์

ตัวอย่าง โปรแกรม 8.15 ใช้สำหรับพิมพ์ภาพจากจอภาพ ที่ได้จากไฟล์ PUPPY.GTP ภาพที่ได้จากการพิมพ์จะมีความสูงเป็นครึ่งหนึ่งของภาพปกติ โปรแกรมนี้เลือกพิมพ์ภาพเฉพาะแถวที่เป็นเลขคู่

โปรแกรม 8.15

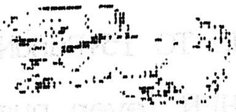
```
PROGRAM PrintScreenLow2;
{print 2 time lower picture }
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE  PictureFromFile(Filename : String);
VAR  GraphDriver,GraphMode        : Integer;
     PicturePointer               : Pointer;
     PictureFile                  : File;
BEGIN
   Assign(PictureFile,Filename);
   Reset(PictureFile);
   GetMem(PicturePointer,FileSize(PictureFile));
   BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
   Close(PictureFile);
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreenL2(StartCol,StartRow,
                            StopCol ,StopRow : Integer);
VAR Row,Col,BandNo      : Integer;
   PicCode              : Byte;
```

```
BEGIN
  BandNo := StartRow;  {top row}
  REPEAT
    Write(Lst,#27#51#23);     {set line spacing 23/216 inch}
    FOR Col := StartCol TO StopCol DO
    BEGIN
      PicCode := 0;                    {blank}
      Write(Lst,#27#42#6#1#0);  {set 90 dot/inch, one column}
      FOR Row := 0+BandNo TO 15+BandNo DO  {read 16 row}
      IF NOT(Odd(Row)) THEN {even number line}
      BEGIN
        IF GetPixel(Col,Row) <> GetBkColor THEN
        CASE Row-BandNo OF
           0 : PicCode := PicCode + 128;
           2 : PicCode := PicCode +  64;
           4 : PicCode := PicCode +  32;
           6 : PicCode := PicCode +  16;
           8 : PicCode := PicCode +   8;
          10 : PicCode := PicCode +   4;
          12 : PicCode := PicCode +   2;
          14 : PicCode := PicCode +   1;
        END;
      END;
      Write(Lst,Chr(PicCode));  {print 1 column}
    END;
    Inc(BandNo,16);
    Write(Lst,#10);          {line feed}
  UNTIL BandNo > StopRow;
  Write(Lst,#12#50);     {form feed & normal spacing}
END;
```

```
BEGIN
   PictureFromFile('PUPPY.GTP');
   Ch := ReadKey;
   IF Ch IN['P','p'] THEN CopyBlockScreenL2(0,0,87,51);
   CloseGraph;
END.
```



รูปที่ 8.20 ภาพจากโปรแกรม 8.15

3. การย่อภาพทั้งในทางสูงและทางกว้าง หมายถึง การพิมพ์ภาพให้แคบลงและต่ำลงกว่าภาพเดิม เพื่อให้ได้ภาพซึ่งมีสัดส่วนตามที่ต้องการ วิธีการพิมพ์ภาพแบบนี้ทำได้โดยการรวมวิธีการย่อภาพทั้งในทางสูงและในทางกว้างเข้าด้วยกัน

ตัวอย่าง โปรแกรม 8.16 ใช้สำหรับพิมพ์ภาพย่อทั้งในทางสูงและในทางกว้าง ซึ่งในโปรแกรมนี้กำหนดย่อความสูงให้เป็นครึ่งหนึ่งของภาพเดิม และย่อความกว้างให้เหลือประมาณครึ่งหนึ่งเช่นเดียวกัน ไฟล์ที่ใช้คือ PUPPY.GTP

โปรแกรม 8.16

```
PROGRAM PrintScreenThinnerLower2;
{print 2 time smaller picture}
USES Crt,Graph,Printer;
VAR Ch : Char;

PROCEDURE PictureFromFile(Filename : String);
VAR GraphDriver,GraphMode   : Integer;
    PicturePointer          : Pointer;
```

```
  PictureFile              : File;
BEGIN
  Assign(PictureFile,Filename);
  Reset(PictureFile);
  GetMem(PicturePointer,FileSize(PictureFile));
  BlockRead(PictureFile,PicturePointer^,FileSize(PictureFile));
  Close(PictureFile);
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  PutImage(0,0,PicturePointer^,NormalPut);
END;

PROCEDURE CopyBlockScreenThinLow(StartCol,StartRow,
                                 StopCol ,StopRow : Integer);
VAR Row,Col,BandNo     : Integer;
   PicCode,N1,N2       : Byte;

BEGIN
  BandNo := StartRow;   {top row}
  N2 := (StopCol - StartCol + 1) DIV 256;
  N1 := (StopCol - StartCol + 1) MOD 256;
  REPEAT
    Write(Lst,#27#51#23);   {set line spacing 23/216 inch}
    Write(Lst,#27#42#3+Chr(N1)+Chr(N2));   {set 240 dot/inch}
    FOR Col := StartCol TO StopCol DO
    BEGIN
      PicCode := 0;                     {blank}
      FOR Row := 0+BandNo TO 15+BandNo DO  {read 16 row}
      BEGIN
        IF NOT(Odd(Row)) THEN
        IF GetPixel(Col,Row) <> GetBkColor THEN
```

```
            CASE Row-BandNo OF
               0 : PicCode := PicCode + 128;
               2 : PicCode := PicCode +  64;
               4 : PicCode := PicCode +  32;
               6 : PicCode := PicCode +  16;
               8 : PicCode := PicCode +   8;
              10 : PicCode := PicCode +   4;
              12 : PicCode := PicCode +   2;
              14 : PicCode := PicCode +   1;
            END;
         END;
         Write(Lst,Chr(PicCode));   {print 1 column}
      END;
      Inc(BandNo,16);
      Write(Lst,#10);              {line feed}
   UNTIL BandNo > StopRow;
   Write(Lst,#12#50);    {form feed & normal spacing}
END;

BEGIN
   PictureFromFile('PUPPY.GTP');
   Ch := ReadKey;
   IF Ch IN['P','p'] THEN CopyBlockScreenThinLow(0,0,87,51);
   CloseGraph
END.
```



รูปที่ 8.21 ภาพจากโปรแกรม 8.16

9

การแสดงและพิมพ์กราฟจากข้อมูลที่กำหนดให้

เราคงจะเคยได้เห็นโปรแกรมสำเร็จรูปประเภทหนึ่ง ซึ่งเมื่อเราป้อนข้อความและตัวเลขเข้าไปแล้ว คอมพิวเตอร์จะแสดงข้อมูลเหล่านั้นออกมาเป็นกราฟ ตัวอย่างโปรแกรมประเภทนี้ที่รู้จักกันแพร่หลายคงจะได้แก่โปรแกรมโลตัส 123

การศึกษาเกี่ยวกับกราฟิกส์จากหลายบทที่ผ่านมา ทำให้เราสามารถนำความรู้เหล่านั้นมาสร้างโปรแกรมแสดงกราฟจากข้อมูลที่กำหนดให้ได้ โปรแกรมที่จะสร้างขึ้นในบทนี้จะขอตั้งชื่อว่า อีซีกราฟ (easygraph-สร้างกราฟได้ง่าย) ซึ่งมีคุณสมบัติดังต่อไปนี้

1. แสดงกราฟได้ 4 ชนิดคือ กราฟจุด กราฟเส้น กราฟแท่ง และ กราฟพาย
2. สามารถป้อนข้อมูลและแก้ไขเปลี่ยนแปลงข้อมูลที่จอภาพได้ด้วยวิธีการง่ายๆ
3. สามารถกำหนดเงื่อนไขแสดงกราฟและเปลี่ยนแปลงได้อย่างสะดวก
4. สามารถบันทึกกราฟไว้ในไฟล์เพื่อการใช้งานในโอกาสต่อไปได้
5. สามารถพิมพ์ภาพกราฟที่ปรากฏบนจอภาพให้ปรากฏบนกระดาษ ด้วยเครื่องพิมพ์ชนิดดอตเมตริกซ์ได้

จากคุณสมบัติดังกล่าวนี้จะเห็นได้ว่า โปรแกรมสามารถอำนวยความสะดวกในการสร้างกราฟให้แก่เราได้ในระดับหนึ่งคือ ทางด้านข้อมูล เราสามารถป้อนข้อมูลเพิ่มลบข้อมูลออก เปลี่ยนแปลงข้อมูลที่มี บันทึกข้อมูลไว้ในไฟล์ และอ่านข้อมูลจากไฟล์มาใช้งานได้ ทางด้านการแสดงกราฟ โปรแกรมสามารถแสดงกราฟได้ 4 ชนิด ทำให้

เรามีโอกาสได้เห็นกราฟต่างชนิดจากข้อมูลชุดเดียวกัน ซึ่งจะช่วยให้เราตัดสินใจเลือกใช้กราฟชนิดที่เหมาะสมกับข้อมูลชุดนั้นได้ สำหรับการกำหนดเงื่อนไขให้แก่กราฟ เช่น การกำหนดความสูงของกราฟ จะมีส่วนช่วยให้ภาพกราฟมีสัดส่วนดีขึ้น ทางด้านการพิมพ์กราฟ โปรแกรมสามารถพิมพ์ภาพจากจอภาพลงบนกระดาษ เพื่อให้สามารถนำไปใช้เป็นเอกสารได้ อย่างไรก็ตาม เนื่องจากอีซีกราฟเป็นโปรแกรมตัวอย่างที่มีขนาดเล็ก การทำงานบางส่วนจึงมีขอบเขตจำกัด ดังรายละเอียดที่จะกล่าวถึงต่อไปนี้

9.1 ชนิดของกราฟ

กำหนดให้อีซีกราฟแสดงกราฟและพิมพ์กราฟได้ 4 ชนิดคือ

9.1.1 กราฟจุด เป็นกราฟชนิดที่ใช้จุดเป็นเครื่องหมายแสดงตำแหน่งที่ข้อมูลในแกน X และแกน Y มีความสัมพันธ์กัน โปรแกรมสามารถแสดงจุดได้ถึง 20 จุดต่อข้อมูล 1 ชุด เช่น แสดงรายได้ของบริษัท A ตั้งแต่ปีพ.ศ. 2511 ถึงปีพ.ศ. 2530 รวม 20 ปี ในเวลาเดียวกันโปรแกรมสามารถแสดงกราฟของข้อมูลได้ถึง 3 ชุด เช่น กราฟของรายได้ รายจ่าย และกำไรในแต่ละปีของบริษัท A

เพื่อให้เห็นความแตกต่างของกราฟจากข้อมูลแต่ละชุด ในโปรแกรมนี้จึงเลือกใช้เครื่องหมายแทนจุดสำหรับข้อมูลแต่ละชุดคือ

เครื่องหมาย + สำหรับข้อมูลชุดที่ 1
เครื่องหมาย * สำหรับข้อมูลชุดที่ 2
เครื่องหมาย x สำหรับข้อมูลชุดที่ 3

โพรซีเยอร์ที่ใช้แสดงกราฟจุดคือ DrawDot ซึ่งมีรายละเอียดดังนี้

```
PROCEDURE DrawDot(VAR DataArray : TypeDataArray;
          GraphTitle, XTitle, YTitle,
          YData1Title, YData2Title, YData3Title,
          YMinString, YMaxString, YStepString : String80);
```

```
CONST
  Y1Chr = '+';
  Y2Chr = '*';
  Y3Chr = 'x';

PROCEDURE DrawKey(YData1Title, YData2Title, YData3Title :
                                                      String80);
BEGIN {DrawKey}
  OutTextXY(WinXMin,GetMaxY - 10, Y1Chr);
  OutTextXY(WinXMin + 14, GetMaxY - 10, YData1Title);
  OutTextXY(WinXMin + 220,GetMaxY - 10, Y2Chr);
  OutTextXY(WinXMin + 234, GetMaxY - 10, YData2Title);
  OutTextXY(WinXMin + 440 ,GetMaxY - 10, Y3Chr);
  OutTextXY(WinXMin + 454, GetMaxY - 10, YData3Title);
END; {DrawKey}

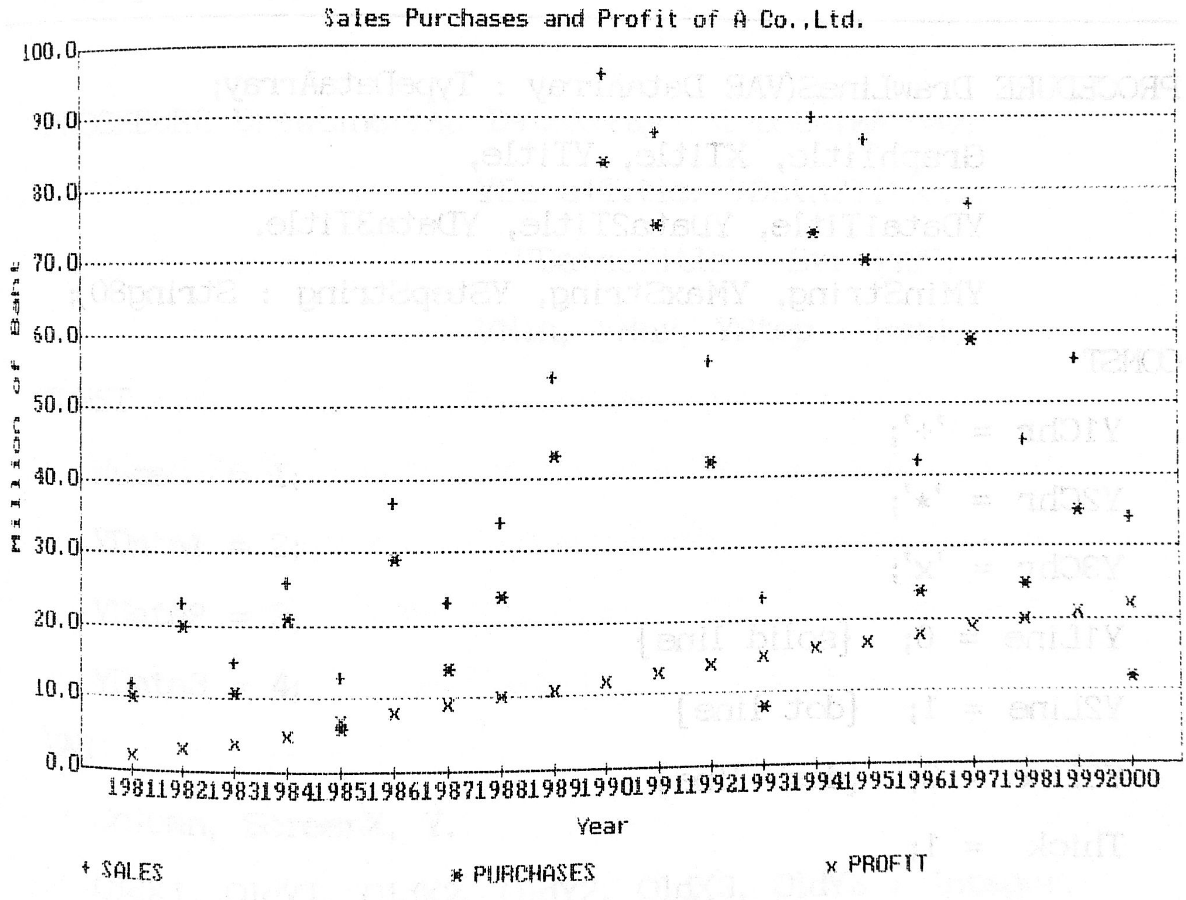
PROCEDURE DrawChrs(VAR DataArray : TypeDataArray;
                       YData1Title, YData2Title,
                        YData3Title : String80;
                       YMin, YMax, YStep : Real);
CONST
  Name   = 1;
  YData1 = 2;
  YData2 = 3;
  YData3 = 4;
VAR
  Column, ScreenX, Y : Integer;
```

```
BEGIN {DrawChrs}
   DrawKey(YData1Title, YData2Title, YData3Title);
   FOR Column := 1 TO ColumnMax(DataArray) DO
   BEGIN
      ScreenX:=WinX(Column +1,0,ColumnMax(DataArray)+1)+10;
      Line(ScreenX+11,WinYMax-2,ScreenX+11,WinYMax+2);
      OutTextXY(ScreenX - ((CharWidth*Length(DataArray
                                    [Column, Name])) DIV 2) + 12,
                  WinYMax + 4, DataArray[Column, Name]);
      Y:=WinY(ValueOf(DataArray[Column,YData1]),YMin,YMax);
      IF Y < WinYMax THEN OutTextXY(ScreenX+8,Y-3,Y1Chr);
      Y:=WinY(ValueOf(DataArray[Column,YData2]),YMin,YMax);
      IF Y < WinYMax THEN OutTextXY(ScreenX+8,Y-3, Y2Chr);
      Y:=WinY(ValueOf(DataArray[Column,YData3]),YMin,YMax);
      IF Y < WinYMax THEN OutTextXY(ScreenX+8,Y-4,Y3Chr);
   END
END; {DrawChrs}

BEGIN {DrawDot}
   YMin  := ValueOf(YMinString);
   YMax  := ValueOf(YMaxString);
   YStep := ValueOf(YStepString);
   OpenGraph;
   Rectangle (WinXMin, WinYMin, WinXMax, WinYMax);
   WriteLabels (GraphTitle, XTitle, YTitle, YMin, YMax, YStep);
   DrawChrs(DataArray,YData1Title,YData2Title,YData3Title,YMin,
         YMax, YStep);
```

```
GetKey(Key, Extended, AllCharSet, AllCharSet, False);
IF Key IN['P','p'] THEN CopyFullScreen;
CloseGraph
END; {DrawDot}
```

ตัวอย่างกราฟจุดที่ได้จากโปรแกรม แสดงในรูปที่ 9.1



รูปที่ 9.1 กราฟรายได้ รายจ่าย และกำไรของบริษัท A โดยแสดงเป็นกราฟจุด

9.1.2 กราฟเส้น เป็นกราฟที่มีลักษณะเดียวกันกับกราฟจุด แต่เพิ่มเติมด้วยการลากเส้นตรงเชื่อมต่อจุดที่อยู่ใกล้กัน เรียงกันตามลำดับจนครบทุกจุดของข้อมูลแต่ละชุด อีซีกราฟสามารถแสดงกราฟเส้นได้พร้อมกันครั้งละ 3 เส้น สำหรับข้อมูล 3

ชุด เช่น เส้นแสดงรายได้ รายจ่าย และกำไร เพื่อให้เห็นความแตกต่างของกราฟจากข้อมูลแต่ละชุดในที่นี้ได้เลือกใช้ลักษณะเส้นสำหรับข้อมูลแต่ละชุด ดังต่อไปนี้

เส้นทึบ ——— สำหรับข้อมูลชุดที่ 1

เส้นจุด .... สำหรับข้อมูลชุดที่ 2

เส้นขีดกลาง ------ สำหรับข้อมูลชุดที่ 3

ไพรซีเยอร์ที่ใช้แสดงกราฟเส้นคือ **DrawLineS** ซึ่งมีรายละเอียดดังนี้

```
PROCEDURE DrawLineS(VAR DataArray : TypeDataArray;
          GraphTitle, XTitle, YTitle,
          YData1Title, YData2Title, YData3Title,
          YMinString, YMaxString, YStepString : String80);
CONST
   Y1Chr = '+';
   Y2Chr = '*';
   Y3Chr = 'x';
   Y1Line = 0;  {solid line}
   Y2Line = 1;  {dot line}
   Y3Line = 3;  {dash line}
   Thick  = 1;

   PROCEDURE DrawKey(YData1Title, YData2Title, YData3Title :
                     String80);
   BEGIN {DrawKey}
      SetColor(Red); SetLineStyle(Y1Line, 0, Thick);
      Line(WinXMin, GetMaxY - 15, WinXMin + 30, GetMaxY - 15);
```

```
  OutTextXY(WinXMin + 14, GetMaxY - 10, YData1Title);
  SetColor(Green); SetLineStyle(Y2Line, 0, Thick);
  Line(WinXMin+220,GetMaxY-15, WinXMin+240+10,GetMaxY-15);
  OutTextXY(WinXMin + 234, GetMaxY - 10, YData2Title);
  SetColor(Blue); SetLineStyle(Y3Line, 0, Thick);
  Line(WinXMin+440,GetMaxY-15,WinXMin+460+10,GetMaxY-15);
  OutTextXY(WinXMin + 454, GetMaxY - 10, YData3Title);
END; {DrawKey}

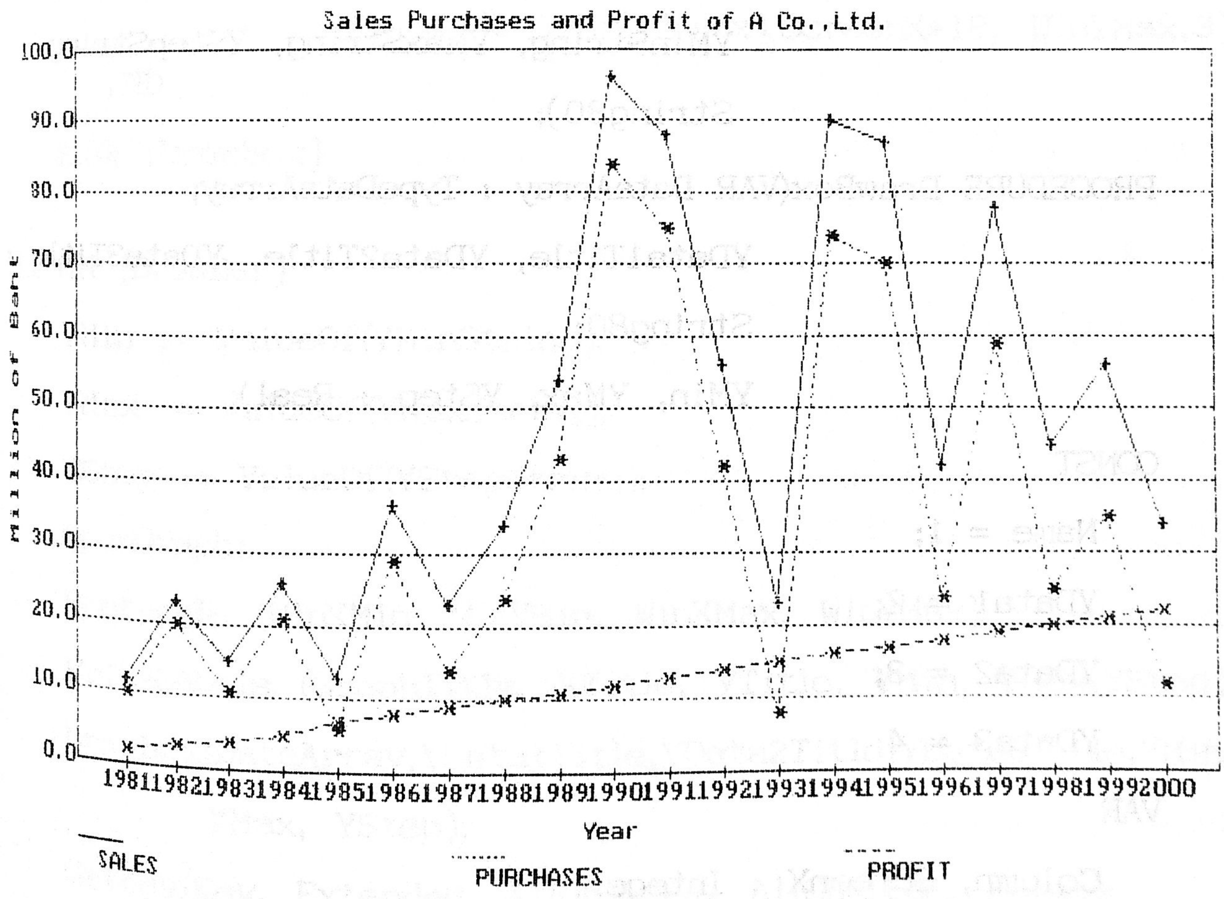
PROCEDURE DrawLine(VAR DataArray : TypeDataArray;
                       YData1Title, YData2Title,
                         YData3Title : String80;
                       YMin, YMax, YStep : Real);
CONST
  Name   = 1;
  YData1 = 2;
  YData2 = 3;
  YData3 = 4;
VAR
  Column, ScreenX, Y,
  OldX1, OldY1, OldX2, OldY2, OldX3, OldY3 : Integer;
BEGIN {DrawLine}
  DrawKey(YData1Title, YData2Title, YData3Title);
  FOR Column := 1 TO ColumnMax(DataArray) DO
  BEGIN
    ScreenX:= WinX(Column+1,0,ColumnMax(DataArray)+1)+10;
    Line(ScreenX+11, WinYMax-2,ScreenX+11, WinYMax+2);
```

```
    OutTextXY(ScreenX - ((CharWidth*Length(DataArray
                                  [Column, Name])) DIV 2) + 12,
              WinYMax + 4, DataArray[Column, Name]);
    SetColor(Red); SetLineStyle(Y1Line, 0, Thick);
    Y:=WinY(ValueOf(DataArray[Column,YData1]),YMin,YMax);
    IF Y < WinYMax THEN OutTextXY(ScreenX+8, Y-3,Y1Chr);
    IF Column > 1 THEN Line(OldX1, OldY1, ScreenX+10, Y);
    OldX1 := ScreenX + 10; OldY1 := Y;
    SetColor(Green); SetLineStyle(Y2Line, 0, Thick);
    Y:=WinY(ValueOf(DataArray[Column,YData2]),YMin,YMax);
    IF Y < WinYMax THEN OutTextXY(ScreenX+8, Y-3, Y2Chr);
    IF Column > 1 THEN Line(OldX2, OldY2, ScreenX+10, Y);
    OldX2 := ScreenX + 10; OldY2 := Y;
    SetColor(Blue); SetLineStyle(Y3Line, 0, Thick);
    Y:=WinY(ValueOf(DataArray[Column,YData3]),YMin,YMax);
    IF Y < WinYMax THEN OutTextXY(ScreenX+8, Y-4, Y3Chr);
    IF Column > 1 THEN Line(OldX3, OldY3, ScreenX+11, Y);
    OldX3 := ScreenX + 10; OldY3 := Y;
  END
END; {Line}

BEGIN {DrawLine}
  YMin  := ValueOf(YMinString);
  YMax  := ValueOf(YMaxString);
  YStep := ValueOf(YStepString);
  OpenGraph;
  Rectangle (WinXMin, WinYMin, WinXMax, WinYMax);
```

```
    WriteLabels (GraphTitle, XTitle, YTitle, YMin, YMax,YStep);
    DrawLine(DataArray, YData1Title, YData2Title, YData3Title,
           YMin, YMax, YStep);
    GetKey(Key, Extended, AllCharSet, AllCharSet, False);
    IF Key IN['P','p'] THEN CopyFullScreen;
    CloseGraph
END; {DrawLine}
```

ตัวอย่างกราฟเส้นที่ได้จากโปรแกรม แสดงในรูปที่ 9.2



รูปที่ 9.2 กราฟรายได้ รายจ่าย และกำไรของบริษัท A โดยแสดงเป็นกราฟเส้น

9.1.3 กราฟแท่ง เป็นกราฟที่มีลักษณะคล้ายกับกราฟจุด แต่เพิ่มเติมด้วยการเขียนรูปแท่งสี่เหลี่ยมจากแต่ละจุดมายังแกน X เนื่องจากอีซีกราฟสามารถแสดงกราฟจากข้อมูล 3 ชุดได้พร้อมกัน โดยแต่ละชุดแสดงด้วยรูปสี่เหลี่ยมต่างกัน ดังนั้นจึงต้องจัดให้รูปสี่เหลี่ยม 3 รูปอยู่เรียงต่อกัน จำนวนรูปแท่งสี่เหลี่ยมของข้อมูล 1 ชุด มีจำนวนเท่ากับกราฟจุดคือ มีได้ถึง 20 แท่ง รวมข้อมูลทั้ง 3 ชุด จะมีได้ถึง 60 แท่ง

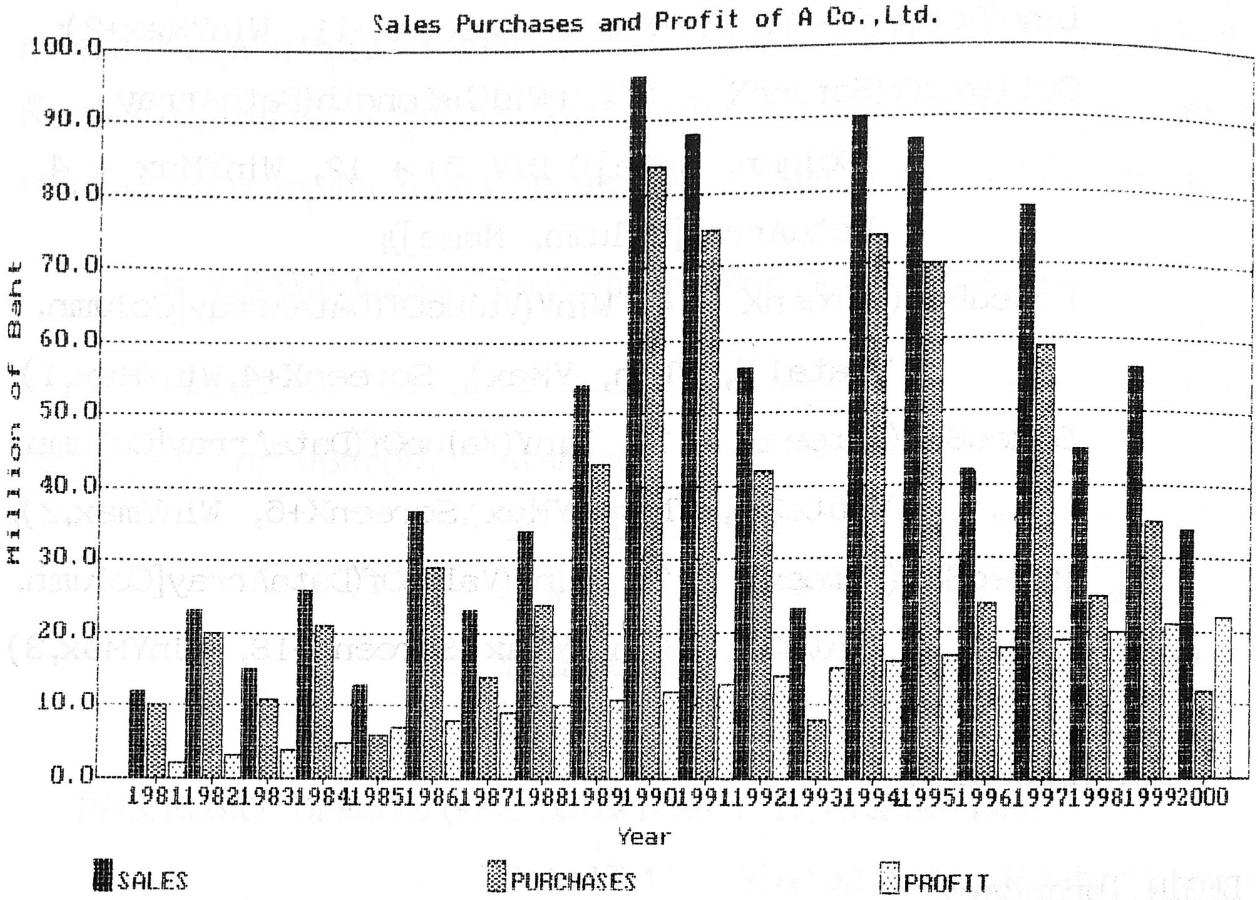
โพรซีเยอร์ที่ใช้แสดงกราฟแท่งคือ **DrawBar** ซึ่งมีรายละเอียดดังนี้

```
PROCEDURE DrawBar(VAR DataArray : TypeDataArray;
                      GraphTitle, XTitle, YTitle,
                      YData1Title, YData2Title, YData3Title,
                      YMinString, YMaxString, YStepString :
                      String80);
  PROCEDURE DrawBox(VAR DataArray : TypeDataArray;
                      YData1Title, YData2Title, YData3Title :
                      String80;
                      YMin, YMax, YStep : Real);
  CONST
    Name = 1;
    YData1 = 2;
    YData2 = 3;
    YData3 = 4;
  VAR
    Column, ScreenX : Integer;
  BEGIN {DrawBox}
    DrawKey(YData1Title, YData2Title, YData3Title);
    FOR Column := 1 TO ColumnMax(DataArray) DO
```

```
    BEGIN
      ScreenX:= WinX(Column+1,0,ColumnMax(DataArray)+1)+10;
      Line(ScreenX+11, WinYMax-2, ScreenX+11, WinYMax+2);
      OutTextXY(ScreenX - ((CharWidth*Length(DataArray
                  [Column, Name])) DIV 2) + 12, WinYMax + 4,
                  DataArray[Column, Name]);
      BoxedBar(ScreenX - 4, WinY(ValueOf(DataArray[Column,
                 YData1]), YMin, YMax), ScreenX+4,WinYMax,1);
      BoxedBar(ScreenX + 16, WinY(ValueOf(DataArray[Column,
                 YData2]), YMin, YMax),ScreenX+6, WinYmax,2);
      BoxedBar(ScreenX + 28, WinY(ValueOf(DataArray[Column,
                 YData3]), YMin, Ymax),ScreenX+18, WinYMax,3)
    END
  END; {DrawBox}

BEGIN {DrawBar}
  YMin := ValueOf(YMinString);
  YMax := ValueOf(YMaxString);
  YStep := ValueOf(YStepString);
  OpenGraph;
  Rectangle (WinXMin, WinYMin, WinXMax, WinYMax);
  WriteLabels (GraphTitle, XTitle, YTitle, YMin, YMax,YStep);
  DrawBox(DataArray,YData1Title,YData2Title,YData3Title,YMin,
        YMax, YStep);
  GetKey(Key, Extended, AllCharSet, AllCharSet, False);
  IF Key IN['P','p'] THEN CopyFullScreen;
  CloseGraph
END; {DrawBar}
```

ตัวอย่างกราฟแท่งที่ได้จากโปรแกรม แสดงในรูปที่ 9.3



รูปที่ 9.3 กราฟรายได้ รายจ่าย และกำไรของบริษัท A โดยแสดงเป็นกราฟแท่ง

9.1.4 กราฟพาย มีลักษณะเป็นภาพวงกลมที่ถูกแบ่งออกเป็นส่วนๆ ด้วยเส้นรัศมี โดยให้แต่ละส่วนแทนขนาดของข้อมูล 1 รายการ และทั้งวงกลมแทนข้อมูล 1 ชุด ดังนั้นกราฟพายจึงแสดงกราฟของข้อมูลได้เพียงครั้งละ 1 ชุด อีซีกราฟสามารถแสดงกราฟของข้อมูลได้ถึง 20 รายการ โปรแกรมจะคำนวณหาขนาดส่วนของวงกลมให้กับข้อมูลแต่ละรายการ โดยกำหนดให้ทั้งวงกลมมี 360 ส่วน แล้วคำนวณค่าแต่ละส่วนของข้อมูลด้วยวิธีการต่อไปนี้

สมมติให้มีข้อมูล 20 รายการคือ n1, n2, ..., n20

Nรวม = n1 + n2 + ,..., + n20

Nรวม = 360

n1 = 360 x n1 / Nรวม

.

.

.

n20 = 360 x n20 / Nรวม

นำค่าของ n1 ถึง n20 ที่ได้จากการคำนวณมากำหนดให้เป็นค่าพารามิเตอร์ของโพรซีเยอร์ **PieSlice** ดังต่อไปนี้

PieSlice(X,Y : Integer; StAngle,Endangle,Radius : Word)

ข้อมูลรายการ n1 กำหนดให้ StAngle = 0
และ EndAngle = n1

ข้อมูลรายการ n2 กำหนดให้ StAngle = n1
และ EndAngle = n2

.

.

.

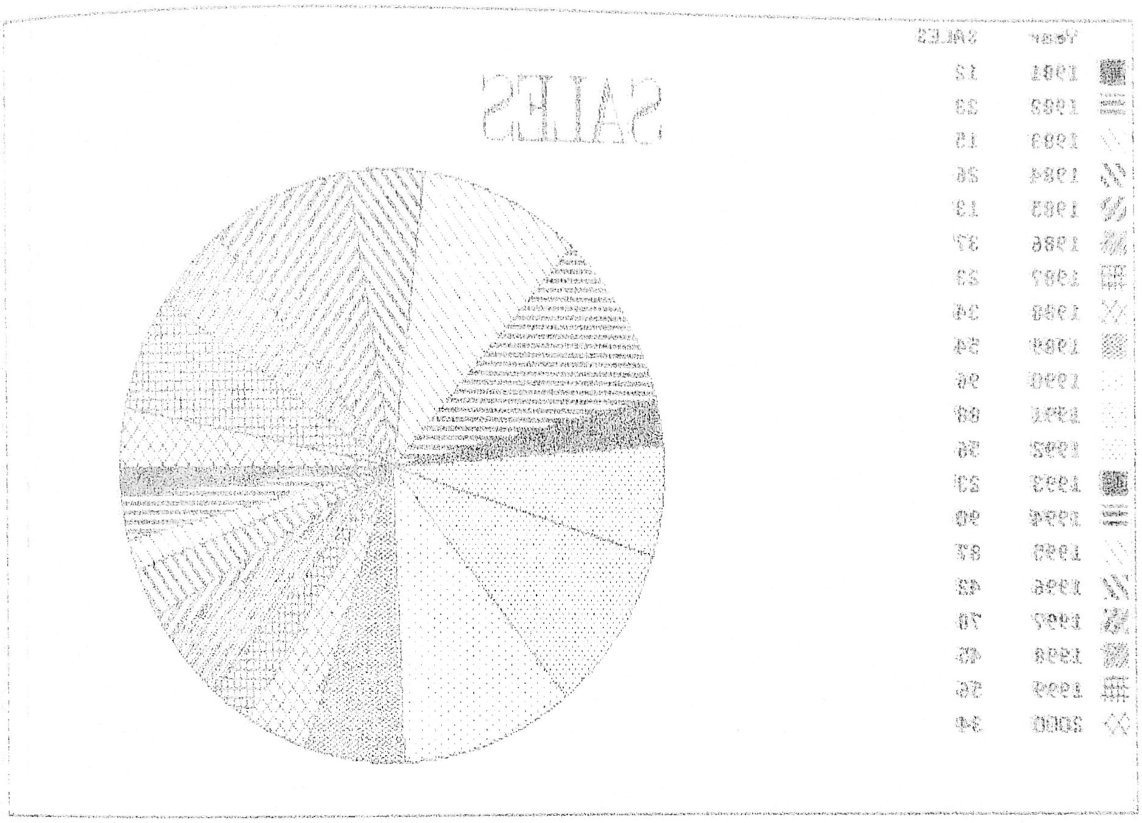
ข้อมูลรายการ n20 กำหนดให้ StAngle = n19
และ EndAngle = n20

แต่ละส่วนของวงกลมจะทำให้เห็นความแตกต่าง โดยการกำหนดสีและลายให้ต่างกันด้วยโพรซีเยอร์ SetFillStyle ตามวิธีที่เราได้เคยศึกษามาแล้ว

โพรซีเยอร์ที่ใช้แสดงกราฟพายคือ **DrawPie** ซึ่งมีรายละเอียดดังนี้

```
PROCEDURE DrawPie( VAR DataArray : TypeDataArray;
                   GraphTitle, XTitle, YData1Title :
                   String80);
VAR Angle  :  ARRAY[0..20] OF Integer;
```

๙.๔ ตัวอย่างการพิมพ์กราฟวงกลม แสดงผลดังรูปที่ ๙.๔



รูปที่ ๙.๔ การพิมพ์กราฟวงกลมแสดงยอดขาย ๒๐ ปีของบริษัท A โดยแสดงเป็นกราฟพาย

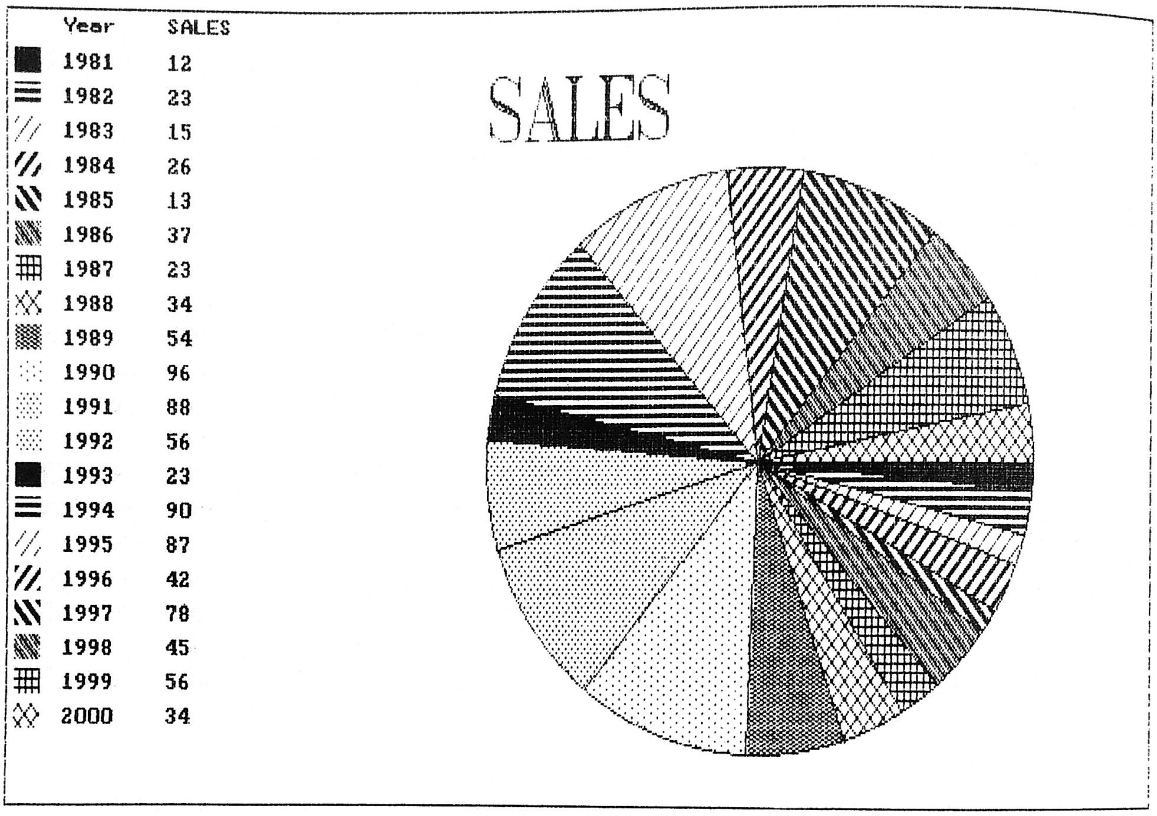
๙.๒ การป้อนข้อมูลและการแสดงผลแบบโต้ตอบ

โปรแกรมที่ใช้ จะทำการป้อนข้อมูลจากแป้นพิมพ์และแสดงผลบนจอภาพ

(screen editor) ซึ่งจะทำการป้อนข้อมูลผ่านทางจอภาพ และแสดงผลบนจอ

ข้อมูลที่ป้อนได้จะแสดงผลบนจอภาพ ดังรูปที่ ๙.๕

ตัวอย่างกราฟพายที่ได้จากโปรแกรม แสดงในรูปที่ 9.4



รูปที่ 9.4 กราฟรายได้แต่ละปีในรอบ 20 ปีของบริษัท A โดยแสดงเป็นกราฟพาย

9.2 การป้อนข้อมูลและการเปลี่ยนแปลงแก้ไขข้อมูล

โปรแกรมอีซีกราฟ จะดำเนินการกับข้อมูลบนจอภาพในลักษณะสกรีนเอดิเตอร์ (screen editor) นั่นคือเราสามารถป้อนข้อมูลเพิ่ม ลบข้อมูลออก และเปลี่ยนแปลงข้อมูลที่มีอยู่ได้ในจอภาพเดียวกัน ลักษณะจอภาพสำหรับป้อนข้อมูลแสดงในรูปที่ 9.5

Group	X_Label	Y1_Data	Y2_Data	Y3_Data
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

↑ → ← ↓ _MoveHighlight HOME_FirstColumn ENTER_EndField ESC_ClearInputLine
BACKSPACE, DEL_DeleteCharacter F10_BackToMenu

รูปที่ 9.5 จอภาพสำหรับป้อนข้อมูลของอีซีกราฟ

จอภาพแบ่งเป็น 2 ส่วนคือ ส่วนบนใช้สำหรับการป้อนข้อมูล แก้ไขข้อมูล และส่วนล่างเป็นบริเวณแสดงหน้าที่ของคีย์ที่ใช้สำหรับการควบคุมซึ่งมีดังนี้

ความหมายของข้อความในบริเวณจอภาพ สำหรับกำหนดเงื่อนไขแสดงกราฟมีดังนี้

ข้อความ	ความหมาย
Graph Title	ชื่อกราฟ
X Axis Title	ชื่อข้อมูลทางแกน X
Y Axis Title	ชื่อข้อมูลทางแกน Y
Y1 Data Title	ชื่อข้อมูลชุดที่ 1 ทางแกน Y
Y2 Data Title	ชื่อข้อมูลชุดที่ 2 ทางแกน Y
Y3 Data Title	ชื่อข้อมูลชุดที่ 3 ทางแกน Y
Minimum Y Axis Range	ค่าต่ำสุดทางแกน Y
Maximum Y Axis Range	ค่าสูงสุดทางแกน Y
Y Axis Step	ขนาดการเพิ่มค่าทางแกน Y

จอภาพสำหรับกำหนดเงื่อนไขแสดงกราฟ มีการทำงานแบบสกรีนเอดิเตอร์เช่นเดียวกันกับโพรซีเยอร์ EditData โพรซีเยอร์กำหนดเงื่อนไขแสดงกราฟคือ โพรซีเยอร์ **DefineGraph** (โปรดดูรายละเอียดในโปรแกรม 9.1)

9.4 การบันทึกกราฟและการอ่านกราฟในไฟล์

โปรแกรมอีซีกราฟไม่ได้บันทึกภาพกราฟไว้ในไฟล์โดยตรง แต่จะบันทึกเฉพาะข้อมูลเกี่ยวกับกราฟ ซึ่งเป็นวิธีการที่เหมาะสมสำหรับโปรแกรมแสดงกราฟซึ่งสามารถแสดงกราฟได้หลายชนิดพร้อมกัน ข้อมูลที่บันทึกไว้ในไฟล์ เป็นข้อมูลที่ได้จากการป้อนข้อมูลตามที่กล่าวแล้วในข้อ 9.2 และ 9.3

9.4.1 การบันทึกกราฟเข้าไฟล์ เมื่อสั่งบันทึกข้อมูล โปรแกรมจะให้ป้อนชื่อไฟล์ดังนี้

```
Enter File name to Save :_
```

เมื่อได้รับชื่อไฟล์แล้ว โปรแกรมจะตรวจสอบว่ามีไฟล์ชื่อนั้นอยู่ในดิสก์หรือไม่ ถ้าไม่มีโปรแกรมจะสร้างไฟล์และบันทึกข้อมูลทันที แต่ถ้าพบว่ามีชื่อไฟล์ซ้ำกัน โปรแกรมจะแสดงข้อความให้เลือกดังนี้

File (ชื่อไฟล์) already exist. Replace Y/N _

ถ้าเลือก Y โปรแกรมจะบันทึกข้อมูลใหม่แทนที่ข้อมูลเดิมที่เคยมีอยู่ในไฟล์

ถ้าเลือก N หรืออักษรอื่น จะไม่มีการบันทึกข้อมูลเข้าในไฟล์นั้น เพราะแสดงว่าผู้ใช้โปรแกรมต้องการตั้งชื่อไฟล์เป็นอย่างอื่น เพื่อไม่ให้ซ้ำกับชื่อไฟล์ที่มีอยู่

โพรซีเยอร์ที่ใช้สำหรับบันทึกกราฟคือ **SaveGraph** ซึ่งมีรายละเอียดดังนี้

```
PROCEDURE SaveGraph(   GraphTitle, XTitle, YTitle,
                       YData1Title, YData2Title, YData3Title,
                       YMin, YMax, YStep : String80;
                   VAR DataArray : TypeDataArray);
VAR
  GraphFile     : Text;
  FileName      : String80;
  Ch            : Char;

  PROCEDURE WriteFile;
  VAR     Row, Column   : Integer;
  BEGIN
     ReWrite(GraphFile);
     WriteLn(GraphFile, GraphTitle);
     WriteLn(GraphFile, XTitle);
     WriteLn(GraphFile, YTitle);
```

[illegible]

File (ชื่อแฟ้ม) already exist! Replace Y/N _

[illegible]

[illegible]

[illegible] SaveGraph [illegible]

```
PROCEDURE SaveGraph(    GraphTitle, XTitle, YTitle,
                        YData1Title, YData2Title, YData3Title,
                        YMin, YMax, YStep : String80;
                        VAR DataArray : TypeDataArray);
VAR
   GraphFile      : Text;
   FileName       : String80;
   Ch             : Char;

PROCEDURE WriteFile;
VAR     Row, Column  : Integer;
BEGIN
   ReWrite(GraphFile);
   WriteLn(GraphFile,   GraphTitle);
   WriteLn(GraphFile,   XTitle);
   WriteLn(GraphFile,   YTitle);
```

```
VAR
   Row, Column : Integer;
   FileName    : String80;
   GraphFile   : Text;
BEGIN {LoadGraph}
   GoToXY(1, 25); ClrEol;
   GoToXY(1,24); ClrEol;
   Write('Enter File name to Load : ');
   ReadLn(FileName);
   IF Length(FileName) > 0 THEN
   BEGIN
      Assign(GraphFile, FileName);
      {$I-} Reset(GraphFile); {$I+}
      IF IoResult = 0 THEN
      BEGIN
         ReadLn(GraphFile, GraphTitle);
         ReadLn(GraphFile, XTitle);
         ReadLn(GraphFile, YTitle);
         ReadLn(GraphFile, YData1Title);
         ReadLn(GraphFile, YData2Title);
         ReadLn(GraphFile, YData3Title);
         ReadLn(GraphFile, YMin);
         ReadLn(GraphFile, YMax);
         ReadLn(GraphFile, YStep);
         FOR Row := 1 TO 20 DO
         FOR Column := 1 TO 4 DO
         ReadLn(GraphFile, DataArray[Row, Column]);
```

```
      Close(GraphFile)
    END
    ELSE
    BEGIN
      Write(#7);
      GoToXY(1,24); ClrEol;
      Write('File ',FileName,' not found. Press ENTER ');
      ReadLn
    END
  END
END; {LoadGraph}
```

9.5 การพิมพ์กราฟ

โปรแกรมอีซีกราฟมีวิธีพิมพ์กราฟ โดยวิธีการก๊อบปี้ภาพทั้งจอภาพไปพิมพ์บนกระดาษเครื่องพิมพ์ที่ใช้ได้เป็นเครื่องพิมพ์ชนิดดอตเมตริกซ์ ยี่ห้อ EPSON หรือเครื่องที่มีคุณสมบัติเหมือนกัน

โพรซีเยอร์ที่ใช้สำหรับพิมพ์กราฟคือ **CopyFullScreen** ซึ่งมีรายละเอียดดังนี้

```
PROCEDURE CopyFullScreen;
VAR Row,Col,BandNo      : Integer;
   PicCode,N1,N2        : Byte;
BEGIN
  N2 := GetMaxX DIV 256;
  N1 := GetMaxX MOD 256;
```

```
                  Close(GraphFile)
                END
              ELSE
              BEGIN
                Write(#7);
                GoToXY(1,24); ClrEol;
                Write('File ',FileName,' not found. Press ENTER ');
                ReadLn
              END
            END
END; {LoadGraph}
```

9.5 การพิมพ์ภาพ

โปรแกรมนี้จะทำการพิมพ์ภาพ โดยจะทำการคัดลอกภาพทั้งจอภาพไปพิมพ์บนกระดาษเครื่องพิมพ์ที่ใช้เป็นเครื่องพิมพ์ชนิดดอตแมทริกซ์ ยี่ห้อ EPSON หรือเครื่องที่มีคุณสมบัติเหมือนกัน

โพรซีเยอร์ที่ใช้สำหรับพิมพ์ภาพคือ CopyFullScreen ซึ่งมีรายละเอียดดังนี้

```
PROCEDURE CopyFullScreen;
VAR Row,Col,BandNo    : Integer;
    PicCode,N1,N2     : Byte;
BEGIN
  N2 := GetMaxX DIV 256;
  N1 := GetMaxX MOD 256;
```

```
Group   X_Label        Y1_Data        Y2_Data        Y3_Data
      ------------------------------------------------------------
   1 ¦
   2 ¦
   3 ¦
   4 ¦
   5 ¦
   6 ¦
   7 ¦
   8 ¦
   9 ¦
  10 ¦
  11 ¦
  12 ¦
  13 ¦
  14 ¦
  15 ¦
  16 ¦
  17 ¦
  18 ¦
  19 ¦
  20 ¦

F2_Save   F3_Load   F4_EditDefinition   F5_EditData   F6_Dot   F7_Line
F8_Bar   F9_Pie   F10_Exit
```

รูปที่ 9.8 จอภาพเมนูของอีซีกราฟ

เมนูคือ สองบรรทัดทางด้านล่างของจอภาพ ซึ่งแต่ละรายการมีความหมายดังนี้

รายการ	ความหมาย
F2_Save	บันทึกข้อมูลกราฟที่อยู่บนจอภาพไว้ในไฟล์
F3_Load	อ่านข้อมูลกราฟในไฟล์มาแสดงบนจอภาพ
F4_EditDefinition	ป้อนและแก้ไขข้อมูลเกี่ยวกับเงื่อนไขการแสดงกราฟ
F5_EditData	ป้อนและแก้ไขข้อมูลกราฟ
F6_Dot	แสดงกราฟจุดจากข้อมูลที่ปรากฏบนจอภาพ
F7_Line	แสดงกราฟเส้นจากข้อมูลที่ปรากฏบนจอภาพ
F8_Bar	แสดงกราฟแท่งจากข้อมูลที่ปรากฏบนจอภาพ
F9_Pie	แสดงกราฟพายจากข้อมูลที่ปรากฏบนจอภาพ
F10_Exit	เลิกใช้โปรแกรมอีซีกราฟ

การเลือกรายการจากเมนูทำได้โดยการกดคีย์ที่ต้องการ ซึ่งโปรแกรมจะทำงานตามรายการที่ถูกเลือกทันที

9.7 โปรแกรมอีซีกราฟ

โปรแกรมอีซีกราฟทั้งโปรแกรม มีรายละเอียดดังต่อไปนี้

โปรแกรม 9.1

```
PROGRAM EasyGraph;
USES Crt,Graph,Printer,GrDriver,GrFont;
CONST
   AllCharSet   = [#8, #13, #27, #32..#126];
   ExtendedSet  = [#60..#68, #71, #72, #75, #77, #80, #83];
   NumSet       = ['0'..'9', '.', '+', '-', #8, #13, #27];
   F2
                = 60;  { save data }
   F3
                = 61;  { load data }
   F4
                = 62;  { edit grapg definition }
```

```
  F5            = 63;  { edit data }
  F6            = 64;  { display graph }
  F7            = 65;  { pie graph }
  F8            = 66;  { dot graph }
  F9            = 67;  { line graph }
  F10           = 68;  { exit }
  Up            = 72;  { up arrow }
  Down          = 80;  { down arrow }
  Right         = 77;  { right arrow }
  Left          = 75;  { left arrow }
  Home          = 71;  { home }
  Del           = 83;  { Del }
  Enter         = -1;
TYPE
  String80         = STRING[80];
  TypeCharacterSet = SET OF Char;
  TypeDataArray    = ARRAY[1..20, 1..4] OF String80;
VAR
  MenuChoice      : Char;
  Done, Extended  : Boolean;
  GraphTitle, XTitle, YTitle, YData1Title, YData2Title, YData3Title,
  YMinStr, YMaxStr, YStepStr : String80;
  YMin, YMax, YStep : Real;
  DataArray          : TypeDataArray;

PROCEDURE OpenGraph;
VAR
  GraphDriver, GraphMode : Integer;
BEGIN {OpenGraph}
  GraphDriver := Detect;
```

```
  Initgraph(GraphDriver,GraphMode, '');
  ClearDevice
END; {OpenGraph}

PROCEDURE EasyTitle;
BEGIN
  OpenGraph;
  {Corners}
  Line(0,3,4,0);
  Line(GetMaxX-4,0,GetMaxX,3);
  Line(0,GetMaxY-3,4,GetMaxY);
  Line(GetMaxX-4,GetMaxY,GetMaxX,GetMaxY-3);
  {Sides}
  Line(4,0,GetMaxX-4,0);
  Line(0,3,0,GetMaxY-3);
  Line(GetMaxX,3,GetMaxX,GetMaxY-3);
  Line(4,GetMaxY,GetMaxX-4,GetMaxY);
  Rectangle(5,4,GetMaxX-5,GetMaxY-80);
  SetTextStyle(TriplexFont,HorizDir,10);
  OutTextXY(200,30,'E');
  OutTextXY(445,30,'SY');
  OutTextXY(71,120,'GR    PH');
  SetUserCharSize(13,1,10,1);
  OutTextXY(235,-48,'A');
  SetTextStyle(SansSerifFont,HorizDir,5);
  OutTextXY(250,280,'Version 1.0');
  REPEAT UNTIL KeyPressed;
  CloseGraph
END;
```

```
PROCEDURE CopyFullScreen;
VAR Row,Col,BandNo    : Integer;
   PicCode,N1,N2      : Byte;
BEGIN
   N2 := GetMaxX DIV 256;
   N1 := GetMaxX MOD 256;
   BandNo := 0;  {top row}
   REPEAT
      Write(Lst,#27#51#23);           {set line spacing 23/216 inch}
      Write(Lst,#27#42#6+Chr(N1)+Chr(N2));   {set 90 dot/inch}
      FOR Col := 0 TO GetMaxX DO        {full X print columns}
      BEGIN
         PicCode := 0;                 {blank}
         FOR Row := 0+BandNo TO 7+BandNo DO
         BEGIN
            IF GetPixel(Col,Row) <> GetBkColor THEN
            CASE Row-BandNo OF
               0 : PicCode := PicCode + 128;
               1 : PicCode := PicCode + 64;
               2 : PicCode := PicCode +  32;
               3 : PicCode := PicCode +  16;
               4 : PicCode := PicCode +   8;
               5 : PicCode := PicCode +   4;
               6 : PicCode := PicCode +   2;
               7 : PicCode := PicCode +   1;
            END;
         END;
         Write(Lst,Chr(PicCode));     {print 1 column}
      END;
      Write(Lst,#10);                 {line feed}
```

```
    Inc(BandNo,8);
  UNTIL BandNo > GetMaxY;              {full Y row}
  Write(Lst,#12#50);                   {form feed & normal spacing}
END;

PROCEDURE GetKey(VAR Key : Char;
             VAR Extended : Boolean;
                 Acceptable,
                 ExtendedAcceptable : TypeCharacterSet;
                 CapsOn : Boolean);
CONST
  ExtendedKey = #0;

  FUNCTION CharOk(Key        : Char;
             Extended : Boolean;
             Acceptable,
             ExtendedAcceptable : TypeCharacterSet) : Boolean;
  BEGIN {CharOk}
    CharOk := (Extended AND (Key in ExtendedAcceptable)) OR
              ((NOT Extended) AND (Key in Acceptable))
  END; {CharOk}

BEGIN {GetKey}
  REPEAT
    Key := Readkey;
    Extended := (Key = ExtendedKey);
    IF Extended THEN Key := ReadKey
              ELSE IF CapsOn THEN Key := UpCase(Key);
    IF NOT CharOk(Key, Extended, Acceptable, ExtendedAcceptable)
      THEN Write(#7)
```

```
   UNTIL CharOk(Key, Extended, Acceptable, ExtendedAcceptable)
END; {GetKey}

PROCEDURE GetField(VAR Field : String80;
                      MaxLength, X, Y : Integer;
                      Acceptable,
                      ExtendedAcceptable : TypeCharacterSet;
                      CapsOn : Boolean;
                   VAR Result : Integer);
CONST
   Beep      = #7;
   Backspace = #8;
   Enter     = #13;
   Escape    = #27;
   Blank     = #176; { space of data input }
   Del       = #83;
VAR
   Cursor   : Integer;
   Key      : Char;
   Extended : Boolean;

   PROCEDURE DrawField(Field : String80;
                       MaxLength, X, Y : Integer);
   VAR
      Space : Integer;
   BEGIN {DrawField}
      GoToXY(X, Y);
      Write(Field);
      FOR Space := 1 TO (MaxLength - Length(Field)) DO Write(Blank)
   END; {DrawField}
```

```
PROCEDURE InitField(VAR Field  : String80;
                    VAR Cursor : Integer;
                        MaxLength, X, Y : Integer);
BEGIN {InitField}
  Field  := '';
  Cursor := 0;
  DrawField(Field, MaxLength, X, Y)
END; {InitField}

PROCEDURE AddKey(VAR Field  : String80;
                 VAR Cursor : Integer;
                     MaxLength : Integer;
                     Key : Char);
BEGIN {AddKey}
  IF (Cursor = MaxLength)
  THEN Write(Beep)
  ELSE
    BEGIN
      Write(Key);
      IF (Cursor = Length(Field)) OR (Length(Field) = 0)
      THEN Field := Field + Key ELSE Field[Cursor + 1] := Key;
      Cursor := Cursor + 1
    END
END; {AddKey}

PROCEDURE DeletePreviousChar(VAR Field  : String80;
                             VAR Cursor : Integer;
                                 MaxLength : Integer);
BEGIN {DeletePreviousChar}
  IF (Cursor > 0) THEN
```

```
    BEGIN
      Write(Backspace, copy(Field, Cursor + 1, MaxLength), Blank);
      Delete(Field, Cursor, 1);
      Cursor := Cursor - 1
    END
    ELSE Write(Beep)
  END; {DeletePreviousChar}

  PROCEDURE DeleteCurrentChar(VAR Field  : String80;
                              VAR Cursor : Integer;
                                 MaxLength : Integer);
  BEGIN {DeleteteCurrentChar}
    IF (Cursor < Length(Field)) THEN
    BEGIN
      Write(copy(Field, Cursor + 2, MaxLength), Blank);
      delete(Field, Cursor + 1, 1)
    END
    ELSE Write(Beep)
  END; {DeteleCurrentChar}

  PROCEDURE ClearBlanks(VAR Field : String80;
                        MaxLength, X, Y : Integer);
  VAR
    Space : Integer;
  BEGIN {ClearBlanks}
    WHILE (Copy(Field, Length(Field), 1) = ' ') DO
    Delete(Field, Length(Field), 1);
    GoToXY(X + Length(Field), Y);
    FOR Space := 1 TO (MaxLength - Length(Field)) DO Write(' ')
  END; {ClearBlanks}
```

```
BEGIN {GetField}
  Cursor := 0;
  DrawField(Field, MaxLength, X, Y);
  REPEAT
    GoToXY(X + Cursor, Y);
    GetKey(Key, Extended, Acceptable, ExtendedAcceptable, CAPSOn);
    IF Extended
    THEN IF (Key = Del)
         THEN DeleteCurrentChar(Field, Cursor, MaxLength)
         ELSE Result := Ord(Key)
    ELSE
      CASE Key OF
        Enter     : Result := -1;
        Backspace : DeletePreviousChar(Field, Cursor, MaxLength);
        Escape    : InitField(Field, Cursor, MaxLength, X, Y)
      ELSE
        AddKey(Field, Cursor, MaxLength, Key)
      END
  UNTIL ((NOT Extended) AND (Key = Enter) OR (Extended) AND
                                               (Key <> Del));
  ClearBlanks(Field, MaxLength, X, Y)
END; {GetField}

PROCEDURE Initialize(VAR Done : Boolean;
                     VAR GraphTitle, XTitle, YTitle,
                         YData1Title, YData2Title, YData3Title,
                         YMin, YMax, YStep : String80;
                     VAR DataArray : TypeDataArray);
VAR
  Row, Column : Integer;
```

```
BEGIN {Initialize}
   Done          := False;
   GraphTitle    := '';
   YData1Title   := '';
   YData2Title   := '';
   YData3Title   := '';
   XTitle        := '';
   YTitle        := '';
   YMin          := '0';
   YMax          := '1';
   YStep         := '1';
   FOR Row       := 1 TO 20 DO
   FOR Column := 1 TO 4 DO DataArray[Row, Column] := ''
END; {Initialize}

PROCEDURE DisplayMenu;
BEGIN {DisplayMenu}
   GoToXY(1, 24); ClrEol;
   Write('F2_Save    F3_Load    F4_EditDefinition    F5_EditData',
         '    F6_Dot    F7_Line');
   GoToXY(1,25); ClrEol;
   Write('F8_Bar    F9_Pie    F10_Exit ')
END; {DisplayMenu}

PROCEDURE ChangeReal(VAR RealString : String80;
                      StartXPos, StartYPos : Integer;
                      VAR LastHitKey : Integer);
VAR
   RealNum    : Real;
   ErrorCode  : Integer;
```

```
BEGIN {ChangeReal}
  REPEAT
    GetField(RealString,  10, StartXPos, StartYPos, NumSet,
             ExtendedSet, False, LastHitKey);
    Val(RealString, RealNum, ErrorCode)
  UNTIL (ErrorCode = 0) OR (RealString = '')
END; {ChangeReal}

PROCEDURE ChangeDefineField(    CurrentLine : Integer;
                            VAR LastHitKey : Integer;
                            VAR GraphTitle, XTitle, YTitle,
                                YData1Title, YData2Title, YData3Title,
                                YMin, YMax, YStep : String80);
BEGIN {ChangeDefineField}
  CASE CurrentLine OF
    1 : GetField(GraphTitle, 60, 15, 4, AllCharSet, ExtendedSet, False,
                 LastHitKey);
    2 : GetField(XTitle, 20, 16, 5, AllCharSet, ExtendedSet, False,
                 LastHitKey);
    3 : GetField(YTitle, 20, 16, 6, AllCharSet, ExtendedSet, False,
                 LastHitKey);
    4 : GetField(YData1Title, 10, 17, 7, AllCharSet, ExtendedSet, False,
                 LastHitKey);
    5 : GetField(YData2Title, 10, 17, 8, AllCharSet, ExtendedSet, False,
                 LastHitKey);
    6 : GetField(YData3Title, 10, 17, 9, AllCharSet, ExtendedSet, False,
                 LastHitKey);
    7 : ChangeReal(YMin, 24, 10,  LastHitKey);
    8 : ChangeReal(YMax, 24, 11,  LastHitKey);
```

```
      9 : ChangeReal(YStep, 15, 12,  LastHitKey);
    END
END; {ChangeDefineField}

PROCEDURE DisplayDefineMenu(VAR GraphTitle, XTitle, YTitle,
                               YData1Title, YData2Title, YData3Title,
                               YMin, YMax, YStep : String80);
BEGIN {DisplayDefineMenu}
    ClrScr;
    WriteLn('                GRAPH DEFINITION');
    WriteLn('      ============================================');
    WriteLn;
    WriteLn('Graph Title : ', GraphTitle);
    WriteLn('X Axis Title : ', XTitle);
    WriteLn('Y Axis Title : ', YTitle);
    WriteLn('Y1 Data Title : ', YData1Title);
    WriteLn('Y2 Data Title : ', YData2Title);
    WriteLn('Y3 Data Title : ', YData3Title);
    WriteLn('Minimum Y Axis Range : ', YMin);
    WriteLn('Maximum Y Axis Range : ', YMax);
    WriteLn('Y Axis Step : ', YStep);
    GoToXY(1,24);
    WriteLn(#24,#32,#25,'_MoveHighlight   ',
            'HOME_FirstColumn  ENTER_EndField  ESC_ClearInputLine');
    Write('BACKSPACE, DEL_DeleteCharacter  F10_BackToMenu')
END; {DisplayDefineMenu}

PROCEDURE AdjustLineNum(VAR CurrentLine : Integer;
                        VAR Done : Boolean;
                            LastHitKey : Integer);
```

```
BEGIN {AdjustLineNum}
  CASE LastHitKey OF
    F10          : Done := True;
    Enter, Down  : IF CurrentLine < 9
                   THEN CurrentLine := CurrentLine + 1;
    Up           : IF CurrentLine > 1
                   THEN CurrentLine := CurrentLine - 1
  END
END; {AdjustLineNum}

PROCEDURE DefineGraph(VAR GraphTitle, XTitle, YTitle,
                      YData1Title, YData2Title, YData3Title,
                      YMin, YMax, YStep : String80);
VAR
  LastHitKey, CurrentLine : Integer;
  Done : Boolean;
BEGIN {DefineGraph}
  CurrentLine := 1;
  Done := False;
  DisplayDefineMenu(GraphTitle, XTitle, YTitle,
                    YData1Title, YData2Title, YData3Title,
                    YMin, YMax, YStep);
  REPEAT
    ChangeDefineField(CurrentLine, LastHitKey, GraphTitle,
                      XTitle, YTitle,
                      YData1Title, YData2Title, YData3Title,
                      YMin, YMax, YStep);
    AdjustLineNum(CurrentLine, Done, LastHitKey)
  UNTIL Done
END; {DefineGraph}
```

```
PROCEDURE InputForm;
VAR
  Row : Integer;
BEGIN {InputForm}
  WriteLn('Group    X_Label          Y1_Data          Y2_Data',
          '        Y3_Data');
  WriteLn('    -----------------------------------------',
          '-------------------');
  FOR Row := 1 TO 20 DO WriteLn(Row:3, ' ¦');
  WriteLn;
  WriteLn(#26,#32,#27,#32,#24,#32,#25,' MoveHighlight   ',
         'HOME FirstColumn  ENTER EndField  ESC ClearInputLine');
  Write('BACKSPACE, DEL DeleteCharacter  F10 BackToMenu')
END; {InputForm}

PROCEDURE DisplayData(VAR DataArray : TypeDataArray);
VAR
  Row, Column : Integer;
BEGIN {DisplayData}
  FOR Row := 1 TO 20 DO
  FOR Column := 1 TO 4 DO
  BEGIN
    GoToXY((Column - 1) * 17 + 9, Row + 2);
    Write(DataArray[Row, Column])
  END
END; {DisplayData}

PROCEDURE DataEditScreen(VAR DataArray : TypeDataArray);
BEGIN {DataEditScreen}
  ClrScr;
```

```
  InputForm;
  DisplayData(DataArray)
END; {DataEditScreen}

PROCEDURE ChangeEditField(VAR FieldToChange : String80;
                              CurrentLine,
                              CurrentField : Integer;
                          VAR LastHitKey : Integer);
BEGIN {ChangeEditField}
  IF (CurrentField = 1)
  THEN GetField(FieldToChange, 10, (CurrentField - 1) * 17 + 9,
      CurrentLine + 2, AllCharSet, ExtendedSet, False,LastHitKey)
  ELSE
      ChangeReal(FieldToChange, (CurrentField - 1) * 17 + 9,
      CurrentLine + 2, LastHitKey)
END; {ChangeEditField}

PROCEDURE AdjustLinColNum(VAR CurrentLine,
                              CurrentField : Integer;
                          VAR Done : Boolean;
                              LastHitKey : Integer);
BEGIN {AdjustLinColNum}
  CASE LastHitKey OF
    Enter, Down : IF CurrentLine < 20
                  THEN CurrentLine := CurrentLine + 1;
    Up          : IF CurrentLine > 1
                  THEN CurrentLine := CurrentLine - 1;
    Left        : IF CurrentField > 1
                  THEN CurrentField := CurrentField - 1;
```

```
    Right          : IF CurrentField < 4
                     THEN CurrentField := CurrentField + 1;
    F10            : Done := true
  END
END; {AdjustLinColNum}

PROCEDURE EditData(VAR DataArray : TypeDataArray);
VAR
  Done : Boolean;
  LastHitKey, CurrentLine, CurrentField : Integer;
BEGIN {EditData}
  CurrentLine  := 1;
  CurrentField := 1;
  Done         := False;
  DataEditScreen(DataArray);
  REPEAT
    ChangeEditField(DataArray[CurrentLine, CurrentField],
                    CurrentLine, CurrentField, LastHitKey);
    AdjustLinColNum(CurrentLine, CurrentField, Done, LastHitKey)
  UNTIL Done
END; {EditData}

FUNCTION ValueOf( Str : String80) : Real;
VAR
  Temp      : Real;
  ErrorCode : Integer;
BEGIN {ValueOf}
  Val(Str, Temp, ErrorCode);
  ValueOf := Temp
END; {ValueOf}
```

```
CONST
   WinXMin = 50;
   WinYMin = 15;
   WinXMax = 700;
   WinYMax = 300;
   CharWidth  = 8;
   CharHeight = 8;
VAR
   {YMin, YMax, YStep : Real;}
   Key : Char;
   {Extended : Boolean;}

FUNCTION WinY(   Y, YMin, YMax : Real) : Integer;
BEGIN {WinY}
   IF YMax = YMin THEN YMax := YMin + 1;
   IF YMin > YMax THEN YMax := YMin - (YMin - YMax) + 1;
   WinY := WinYMax - Round((WinYMax - WinYMin) *
           (Y - YMin) / (YMax - YMin))
END; {WinY}

FUNCTION WinX(   X, XMin, XMax : Real) : Integer;
BEGIN {WinX}
   WinX := Round((WinXMax - WinXMin) * (X - XMin) / (XMax - XMin))
END; {WinX}

PROCEDURE WriteLabels(GraphTitle, XTitle, YTitle : String80;
                      YMin, YMax, YStep : Real);
VAR
   Y : Real;
   ScreenY : Integer;
```

```
   YString : String80;
 BEGIN {WriteLabels}
   OutTextXY(((getmaxx+1) DIV 2) -
            ((CharWidth*Length(GraphTitle)) DIV 2), 0, GraphTitle);
   OutTextXY(((getmaxx-1) DIV 2)-
            ((CharWidth*Length(XTitle)) DIV 2), (getmaxy -
              CharHeight - 20), XTitle);
   SetTextStyle(DefaultFont, VertDir, 1);
   OutTextXY(CharWidth, (WinYMax DIV 2) + (WinYMin DIV 2)-
           ((CharWidth*Length (YTitle)) DIV 2), YTitle);
   SetTextStyle(DefaultFont, HorizDir, 1);
   SetLineStyle(DottedLn, 0, NormWidth);
   Y := YMin;
   REPEAT
      ScreenY := WinY(Y, YMin, YMax);
      Str(Y:0:1, YString);
      OutTextXY((WinXMin - (CharWidth*Length(YString))),
               (ScreenY-(CharWidth DIV 2)), YString);
      Line(WinXMin, ScreenY, WinXMax, ScreenY);
      Y := Y + YStep
   UNTIL (Y > YMax);
   SetLineStyle(SolidLn, 0, NormWidth)
END; {WriteLabels}

PROCEDURE BoxedBar( X1, Y1, X2, Y2, YData : Integer);
CONST
  YData1Fill = SolidFill;
  YData2Fill = InterleaveFill;
  YData3Fill = CloseDotFill;
```

```
BEGIN {BoxedBar}
  CASE YData OF
     1 : SetFillStyle(YData1Fill, 1);
     2 : SetFillStyle(YData2Fill, 2);
     3 : SetFillStyle(YData3Fill, 3)
  END;
  Bar(X1, Y1, X2, Y2);
  Rectangle(X1, Y1, X2, Y2)
END; {BoxedBar}

PROCEDURE DrawKey(YData1Title, YData2Title, YData3Title : String80);
BEGIN {DrawKey}
  BoxedBar(WinXMin,GetMaxY - 15, WinXMin + 10, GetMaxY - 3, 1);
  OutTextXY(WinXMin + 14, GetMaxY - 10, YData1Title);
  BoxedBar(WinXMin + 220,GetMaxY - 15,WinXMin + 230,GetMaxY - 3,2);
  OutTextXY(WinXMin + 234, GetMaxY - 10, YData2Title);
  BoxedBar(WinXMin + 440,GetMaxY - 15,WinXMin + 450,GetMaxY - 3, 3);
  OutTextXY(WinXMin + 454, GetMaxY - 10, YData3Title);
END; {DrawKey}

FUNCTION ColumnMax(VAR DataArray : TypeDataArray) : Integer;
VAR
  Column : Integer;
BEGIN {ColumnMax}
  ColumnMax := 0;
  FOR Column := 1 TO 20 DO
    IF (DataArray[Column, 2] <> '') OR
       (DataArray[Column, 3] <> '') OR
       (DataArray[Column, 4] <> '')
    THEN
```

```
      ColumnMax := Column
END; {ColumnMax}

PROCEDURE DrawBar(VAR DataArray : TypeDataArray;
                      GraphTitle, XTitle, YTitle,
                      YData1Title, YData2Title, YData3Title,
                      YMinString,YMaxString,YStepString : String80);
   PROCEDURE DrawBox(VAR DataArray : TypeDataArray;
                      YData1Title,YData2Title,YData3Title : String80;
                      YMin, YMax, YStep : Real);
   CONST
      Name = 1;
      YData1 = 2;
      YData2 = 3;
      YData3 = 4;
   VAR
      Column, ScreenX : Integer;
   BEGIN {DrawBox}
      DrawKey(YData1Title, YData2Title, YData3Title);
      FOR Column := 1 TO ColumnMax(DataArray) DO
      BEGIN
         ScreenX := WinX(Column + 1,0,ColumnMax(DataArray) + 1) + 10;
         Line(ScreenX + 11, WinYMax - 2, ScreenX + 11, WinYMax + 2);
         OutTextXY(ScreenX -
             ((CharWidth*Length(DataArray[Column,Name])) DIV 2) + 12,
                   WinYMax + 4,DataArray[Column, Name]);
        BoxedBar(ScreenX - 4, WinY(ValueOf(DataArray[Column,YData1]),
                   YMin, YMax), ScreenX + 4, WinYMax, 1);
        BoxedBar(ScreenX + 16,WinY(ValueOf(DataArray[Column,YData2]),
                   YMin, YMax), ScreenX + 6, WinYmax, 2);
```

```
    BoxedBar(ScreenX + 28,WinY(ValueOf(DataArray[Column,YData3]),
                YMin, Ymax), ScreenX + 18, WinYMax, 3)
   END
  END; {DrawBox}

BEGIN {DrawBar}
  YMin := ValueOf(YMinString);
  YMax := ValueOf(YMaxString);
  YStep := ValueOf(YStepString);
  OpenGraph;
  Rectangle (WinXMin, WinYMin, WinXMax, WinYMax);
  WriteLabels (GraphTitle, XTitle, YTitle, YMin, YMax, YStep);
  DrawBox (DataArray, YData1Title, YData2Title, YData3Title, YMin,
        YMax, YStep);
  GetKey(Key Extended, AllCharSet, AllCharSet, False);
  IF Key IN['P','p'] THEN CopyFullScreen;
  CloseGraph
END; {DrawBar}

(******************     Dot Graph    ***********************)

PROCEDURE DrawDot(VAR DataArray : TypeDataArray;
                  GraphTitle, XTitle, YTitle,
                  YData1Title, YData2Title, YData3Title,
                  YMinString,YMaxString,YStepString : String80);
CONST
  Y1Chr = '+';
  Y2Chr = '*';
  Y3Chr = 'x';
```

```
PROCEDURE DrawKey(YData1Title, YData2Title, YData3Title : String80);
BEGIN {DrawKey}
   OutTextXY(WinXMin,GetMaxY - 10, Y1Chr);
   OutTextXY(WinXMin + 14, GetMaxY - 10, YData1Title);
   OutTextXY(WinXMin + 220,GetMaxY - 10, Y2Chr);
   OutTextXY(WinXMin + 234, GetMaxY - 10, YData2Title);
   OutTextXY(WinXMin + 440 ,GetMaxY - 10, Y3Chr);
   OutTextXY(WinXMin + 454, GetMaxY - 10, YData3Title);
END; {DrawKey}

PROCEDURE DrawChrs(VAR DataArray : TypeDataArray;
                   YData1Title,YData2Title,YData3Title : String80;
                   YMin, YMax, YStep : Real);
CONST
   Name   = 1;
   YData1 = 2;
   YData2 = 3;
   YData3 = 4;
VAR
   Column, ScreenX, Y : Integer;
BEGIN {DrawChrs}
   DrawKey(YData1Title, YData2Title, YData3Title);
   FOR Column := 1 TO ColumnMax(DataArray) DO
   BEGIN
     ScreenX := WinX(Column + 1, 0, ColumnMax(DataArray) + 1) + 10;
     Line(ScreenX + 11, WinYMax - 2, ScreenX + 11, WinYMax + 2);
     OutTextXY(ScreenX -
           ((CharWidth*Length(DataArray[Column,Name])) DIV 2) + 12,
             WinYMax + 4, DataArray[Column, Name]);
     Y := WinY(ValueOf(DataArray[Column, YData1]), YMin, YMax);
```

```
      IF Y < WinYMax THEN OutTextXY(ScreenX + 8, Y - 3, Y1Chr);
      Y := WinY(ValueOf(DataArray[Column, YData2]), YMin, YMax);
      IF Y < WinYMax THEN OutTextXY(ScreenX + 8, Y - 3, Y2Chr);
      Y := WinY(ValueOf(DataArray[Column, YData3]), YMin, YMax);
      IF Y < WinYMax THEN OutTextXY(ScreenX + 8, Y - 4, Y3Chr);
    END
  END; {DrawChrs}

BEGIN {DrawDot}
  YMin  := ValueOf(YMinString);
  YMax  := ValueOf(YMaxString);
  YStep := ValueOf(YStepString);
  OpenGraph;
  Rectangle (WinXMin, WinYMin, WinXMax, WinYMax);
  WriteLabels (GraphTitle, XTitle, YTitle, YMin, YMax, YStep);
  DrawChrs (DataArray, YData1Title, YData2Title, YData3Title, YMin,
          YMax, YStep);
  GetKey(Key, Extended, AllCharSet, AllCharSet, False);
  IF Key IN['P','p'] THEN CopyFullScreen;
  CloseGraph
END; {DrawDot}

(*****************      Line Graph    **********************)
PROCEDURE DrawLineS(VAR DataArray : TypeDataArray;
                    GraphTitle, XTitle, YTitle,
                    YData1Title, YData2Title, YData3Title,
                    YMinString,YMaxString,YStepString : String80);
CONST
  Y1Chr = '+';
  Y2Chr = '*';
```

```
Y3Chr = 'x';
Y1Line = 0;  {solid line}
Y2Line = 1;  {dot line}
Y3Line = 3;  {dash line}
Thick  = 1;

PROCEDURE DrawKey(YData1Title, YData2Title, YData3Title : String80);
BEGIN {DrawKey}
   SetColor(Red); SetLineStyle(Y1Line, 0, Thick);
   Line(WinXMin, GetMaxY - 15, WinXMin + 30, GetMaxY - 15);
   OutTextXY(WinXMin + 14, GetMaxY - 10, YData1Title);
   SetColor(Green); SetLineStyle(Y2Line, 0, Thick);
   Line(WinXMin + 220,GetMaxY - 15,WinXMin + 240 + 10,GetMaxY - 15);
   OutTextXY(WinXMin + 234, GetMaxY - 10, YData2Title);
   SetColor(Blue); SetLineStyle(Y3Line, 0, Thick);
   Line(WinXMin + 440 ,GetMaxY - 15, WinXMin + 460 + 10,
        GetMaxY - 15);
   OutTextXY(WinXMin + 454, GetMaxY - 10, YData3Title);
END; {DrawKey}

PROCEDURE DrawLine(VAR DataArray : TypeDataArray;
                   YData1Title,YData2Title,YData3Title:String80;
                   YMin, YMax, YStep : Real);
CONST
   Name    = 1;
   YData1 = 2;
   YData2 = 3;
   YData3 = 4;
VAR
   Column, ScreenX, Y,
   OldX1, OldY1, OldX2, OldY2, OldX3, OldY3 : Integer;
```

```
BEGIN {DrawLine}
  DrawKey(YData1Title, YData2Title, YData3Title);
  FOR Column := 1 TO ColumnMax(DataArray) DO
  BEGIN
    ScreenX := WinX(Column + 1, 0, ColumnMax(DataArray) + 1) + 10;
    Line(ScreenX + 11, WinYMax - 2, ScreenX + 11, WinYMax + 2);
    OutTextXY(ScreenX -
        ((CharWidth*Length(DataArray[Column, Name])) DIV 2) + 12,
        WinYMax + 4, DataArray[Column, Name]);
    SetColor(Red); SetLineStyle(Y1Line, 0, Thick);
    Y := WinY(ValueOf(DataArray[Column, YData1]), YMin, YMax);
    IF Y < WinYMax THEN OutTextXY(ScreenX + 8, Y - 3, Y1Chr);
    IF Column > 1 THEN Line(OldX1, OldY1, ScreenX + 10, Y);
    OldX1 := ScreenX + 10; OldY1 := Y;
    SetColor(Green); SetLineStyle(Y2Line, 0, Thick);
    Y := WinY(ValueOf(DataArray[Column, YData2]), YMin, YMax);
    IF Y < WinYMax THEN OutTextXY(ScreenX + 8, Y - 3, Y2Chr):
    IF Column > 1 THEN Line(OldX2, OldY2, ScreenX + 10, Y);
    OldX2 := ScreenX + 10; OldY2 := Y;
    SetColor(Blue); SetLineStyle(Y3Line, 0, Thick);
    Y := WinY(ValueOf(DataArray[Column, YData3]), YMin, YMax);
    IF Y < WinYMax THEN OutTextXY(ScreenX + 8, Y - 4, Y3Chr);
    IF Column > 1 THEN Line(OldX3, OldY3, ScreenX + 11, Y);
    OldX3 := ScreenX + 10; OldY3 := Y;
  END
END; {Line}

BEGIN {DrawLine}
  YMin   := ValueOf(YMinString);
  YMax   := ValueOf(YMaxString);
```

```
  YStep := ValueOf(YStepString);
  OpenGraph;
  Rectangle (WinXMin, WinYMin, WinXMax, WinYMax);
  WriteLabels (GraphTitle, XTitle, YTitle, YMin, YMax, YStep);
  DrawLine(DataArray, YData1Title, YData2Title, YData3Title, YMin,
           YMax, YStep);
  GetKey(Key, Extended, AllCharSet, AllCharSet, False);
  IF Key IN['P','p'] THEN CopyFullScreen;
  CloseGraph
END; {DrawLine}


(*************** Pie **************)
PROCEDURE DrawPie( VAR DataArray : TypeDataArray;
                       GraphTitle, XTitle, YData1Title : String80);
VAR Angle  :  ARRAY[0..20] OF Integer;
    Step, CurrentAngle, Row, Pattern : Integer;
    SumItem, YMin, YMax, YStep : Real;
    Key : Char;
    Extended : Boolean;

  FUNCTION ItemMax(VAR DataArray : TypeDataArray) : Integer;
  VAR
     Item : Integer;
  BEGIN {ItemMax}
     ItemMax := 0;
     FOR Item := 1 TO 20 DO
     IF (DataArray[Item, 2] <> '') THEN ItemMax := Item
  END; {ItemMax}
```

```
BEGIN {DrawPie}
  OpenGraph;
  Rectangle(0, 0, GetMaxX-2, GetMaxY);
  OutTextXY(35, 4, XTitle + '     ' + YData1Title);
  Row := 20;    SumItem := 0;   Pattern := 0;
  FOR Step := 1 TO ItemMax(DataArray) DO
  BEGIN
    IF Pattern < 12 THEN Inc(Pattern) ELSE Pattern := 1; {pattern}
    SetFillStyle(Pattern, Step);
    Bar(5, Row - 2, 20, Row - 2 + 10);
    SumItem  := SumItem + ValueOf(DataArray[Step, 2]);
    OutTextXY(35,Row,DataArray[Step, 1] + '   ' + DataArray[Step, 2]);
    Inc(Row, 15);
  END;
  Pattern := 0;    Angle[0] := 0;
  FOR Step := 1 TO ItemMax(DataArray) DO {calculate an angle}
  BEGIN
    CurrentAngle := Round(360 / SumItem * ValueOf(DataArray[Step,2]));
    Angle[Step]  := Angle[Step - 1] + CurrentAngle;{acumulate angle}
  END;
  IF Angle[Step] > 360 THEN Angle[Step] := 360; {adjust to 360}
  FOR Step := 1 TO ItemMax(DataArray) DO
  BEGIN
    IF Pattern < 12 THEN Inc(Pattern) ELSE Pattern := 1; {pattern}
    SetFillStyle(Pattern, Step);
    PieSlice(GetMaxX - 250, GetMaxY - 150, 360 - Angle[Step],
             360 - Angle[Step - 1], 170);
  END;
  SetTextStyle(TriplexFont, HorizDir, 5);
  OutTextXY(300, 20, YData1Title);    {graph name}
```

```
   GetKey(Key, Extended, AllCharSet, AllCharSet, False);
   IF Key IN['P','p'] THEN CopyFullScreen;
   CloseGraph
END; {DrawPie}

(************** End Pie ***********)

PROCEDURE SaveGraph(     GraphTitle, XTitle, YTitle,
                         YData1Title, YData2Title, YData3Title,
                         YMin, YMax, YStep : String80;
                   VAR DataArray : TypeDataArray);
VAR
   GraphFile      : Text;
   FileName       : String80;
   Ch             : Char;

   PROCEDURE WriteFile;
   VAR    Row, Column  : Integer;
   BEGIN
      ReWrite(GraphFile);
      WriteLn(GraphFile, GraphTitle);
      WriteLn(GraphFile, XTitle);
      WriteLn(GraphFile, YTitle);
      WriteLn(GraphFile, YData1Title);
      WriteLn(GraphFile, YData2Title);
      WriteLn(GraphFile, YData3Title);
      WriteLn(GraphFile, YMin);
      WriteLn(GraphFile, YMax);
      WriteLn(GraphFile, YStep);
      FOR Row := 1 TO 20 DO
```

```
    FOR Column := 1 TO 4 DO
    WriteLn(GraphFile, DataArray[Row, Column]);
  END;

BEGIN {SaveGraph}
  GoToXY(1, 25); ClrEol;
  GoToXY(1,24); ClrEol;
  Write('Enter File name to Save : ');
  ReadLn(FileName);
  IF Length(FileName) > 0 THEN
  BEGIN
    Assign(GraphFile, FileName);
    {$I-} Reset(GraphFile); {$I+}
    IF IoResult = 0 THEN   {file exist}
    BEGIN
      Write(#7);
      GoToXY(1,24); ClrEol;
      Write('File ',Filename,' already exist. Replace Y/N ');
      ReadLn(Ch);
      IF Ch IN['Y','y'] THEN WriteFile
    END
    ELSE
    WriteFile;
    Close(GraphFile)
  END;
END; {SaveGraph}

PROCEDURE LoadGraph(VAR GraphTitle, XTitle, YTitle,
                    YData1Title, YData2Title, YData3Title,
                    YMin, YMax, YStep : String80;
```

```
                         VAR DataArray : TypeDataArray);
VAR
   Row, Column : Integer;
   FileName    : String80;
   GraphFile   : Text;
BEGIN {LoadGraph}
   GoToXY(1, 25); ClrEol;
   GoToXY(1,24); ClrEol;
   Write('Enter File name to Load : ');
   ReadLn(FileName);
   IF Length(FileName) > 0 THEN
   BEGIN
      Assign(GraphFile, FileName);
      {$I-} Reset(GraphFile); {$I+}
      IF IoResult = 0 THEN
      BEGIN
         ReadLn(GraphFile, GraphTitle);
         ReadLn(GraphFile, XTitle);
         ReadLn(GraphFile, YTitle);
         ReadLn(GraphFile, YData1Title);
         ReadLn(GraphFile, YData2Title);
         ReadLn(GraphFile, YData3Title);
         ReadLn(GraphFile, YMin);
         ReadLn(GraphFile, YMax);
         ReadLn(GraphFile, YStep);
         FOR Row := 1 TO 20 DO
         FOR Column := 1 TO 4 DO
         ReadLn(GraphFile, DataArray[Row, Column]);
         Close(GraphFile)
      END
```

```
    ELSE
    BEGIN
      Write(#7);
      GoToXY(1,24); ClrEol;
      Write('File ',FileName,' not found. Press ENTER ');
      ReadLn
    END
  END
END; {LoadGraph}

PROCEDURE DoMenuChoice(      MenuChoice : Char;
                        VAR GraphTitle, XTitle, YTitle,
                            YData1Title, YData2Title, YData3Title,
                            YMin, YMax, YStep : String80;
                        VAR DataArray : TypeDataArray);
BEGIN {DoMenuChoice}
  CASE Ord(MenuChoice) OF
    F2 : SaveGraph(GraphTitle, XTitle, YTitle,
                 YData1Title, YData2Title, YData3Title,
                 YMin,  YMax,  YStep, DataArray);
    F3 : LoadGraph(GraphTitle, XTitle, YTitle,
                 YData1Title, YData2Title, YData3Title,
                 YMin,  YMax,  YStep, DataArray);
    F4 : DefineGraph(GraphTitle,XTitle, YTitle,
                   YData1Title, YData2Title, YData3Title,
                   YMin, YMax, YStep);
    F5 : EditData(DataArray);
    F6 : DrawDot(DataArray, GraphTitle, XTitle, YTitle,
                   YData1Title, YData2Title, YData3Title,
                   YMin, YMax, YStep);
```

```
      F7  : DrawLineS(DataArray, GraphTitle, XTitle, YTitle,
                     YData1Title, YData2Title, YData3Title,
                     YMin, YMax, YStep);
      F8  : DrawBar(DataArray, GraphTitle, XTitle, YTitle,
                     YData1Title, YData2Title, YData3Title,
                     YMin, YMax, YStep);
      F9  : DrawPie(DataArray, GraphTitle, XTitle, YData1Title);
      F10: Done := True
   END
END; {DoMenuChoice}

BEGIN
   Initialize(Done, GraphTitle, XTitle,YTitle,
              YData1Title, YData2Title, YData3Title,
              YMinStr, YMaxStr, YStepStr, DataArray);
   EasyTitle;
   REPEAT
      DataEditScreen(DataArray);
      DisplayMenu;
      GetKey(MenuChoice, Extended, [], ExtendedSet, False);
      DoMenuChoice(MenuChoice, GraphTitle, XTitle,YTitle,
                   YData1Title, YData2Title, YData3Title,
                   YMinStr, YMaxStr, YStepStr, DataArray);
   UNTIL Done;
END.
```

10
การสร้างและแสดงอักษรกราฟิกส์

คอมพิวเตอร์กราฟิกส์ นอกจากจะเกี่ยวข้องกับภาพแล้ว ยังมีความเกี่ยวข้องกับอักษรด้วย ในบทนี้จะกล่าวถึงตัวอักษรกราฟิกส์

10.1 ความหมายของอักษรและฟอนต์

อักษร หมายถึง ตัวหนังสือ ตัวเลขและสัญลักษณ์ต่างๆ ที่เครื่องคอมพิวเตอร์รับรู้ และสามารถแสดงให้เห็นได้ เช่น อักษรที่รับจากคีย์บอร์ด ตัวหนังสือและตัวเลข **ที่**แสดงบนจอภาพ

ฟอนต์ (font) หมายถึง อักษรชุดหนึ่งซึ่งมีลักษณะรูปแบบ และขนาดเดียวกัน

10.2 ประเภทของอักษร

อักษรที่ใช้ในคอมพิวเตอร์แบ่งได้เป็น 2 ประเภทคือ อักษรฮาร์ดแวร์ (hard-ware character) และอักษรซอฟต์แวร์ (software character) ซึ่งมีรายละเอียดดังต่อไปนี้

10.2.1 อักษรฮาร์ดแวร์ เป็นชุดของอักษรหรือฟอนต์ ซึ่งผู้สร้างเครื่องคอมพิวเตอร์ได้สร้างขึ้น และเก็บไว้ในเครื่องคอมพิวเตอร์ โดยเก็บอยู่ในหน่วยความจำ

ประเภท PROM (Programmable Read Only Memory) คอมพิวเตอร์จะแสดงข้อความต่างๆ ให้เห็นที่จอภาพตั้งแต่เริ่มเปิดเครื่องด้วยอักษรประเภทนี้ ถ้าไม่มีคำสั่งเปลี่ยนฟอนต์ คอมพิวเตอร์จะใช้ฟอนต์นี้ตลอดเวลา

ข้อจำกัดของอักษรประเภทนี้คือ

1. มีแบบและขนาดจำกัด ตามที่ได้สร้างเอาไว้ในคอมพิวเตอร์ ตามปกติจะมีเพียงภาษาเดียวคือ ภาษาอังกฤษ ถึงแม้ว่าจะสามารถเพิ่มอักษรภาษาอื่นเข้าไปอีกได้ แต่ก็ยังมีขีดจำกัดทางด้านฮาร์ดแวร์ การสร้างอักษรแบบนี้จึงไม่เป็นที่นิยม

2. การสร้างอักษรมีวิธีการยุ่งยากกว่าและต้องมีความรู้ทางด้านฮาร์ดแวร์ด้วย จึงทำให้ผู้สนใจทางด้านนี้มีจำนวนน้อย

3. แสดงอักษรได้ในลักษณะจำกัด ซึ่งส่วนใหญ่จะแสดงในลักษณะตั้งตรง เราจะเปลี่ยนการแสดงให้เป็นลักษณะอื่น เช่น เอียงขวา เอียงซ้ายไม่ได้

4. ต้องแสดงอักษร ณ ตำแหน่งที่แน่นอนซึ่งนิยมกำหนดตำแหน่งเป็นแถว(row) และคอลัมน์ (column) การแสดงอักษรในตำแหน่งที่แตกต่างไปจากที่กำหนดไว้ เช่น แสดงที่ตำแหน่งครึ่งคอลัมน์ หรือครึ่งแถวจะทำไม่ได้

10.2.2 อักษรซอฟต์แวร์ เป็นอักษรที่เก็บอยู่ในไฟล์ สำหรับให้โปรแกรมต่างๆ เรียกไปใช้ อักษรประเภทนี้แม้จะมีข้อจำกัดในการใช้คือ จะต้องใช้ในโหมดกราฟิกส์เท่านั้น แต่อักษรกราฟิกส์ก็มีส่วนที่ดีกว่าอักษรฮาร์ดแวร์อยู่หลายประการคือ

1. มีหลายแบบ เนื่องจากสร้างอักษรได้ง่ายด้วยโปรแกรมขนาดเล็ก ซึ่งเมื่อได้ศึกษาวิธีใช้โปรแกรมในเวลาเพียงเล็กน้อย ทุกคนก็จะสร้างอักษรขึ้นมาได้ จึงทำให้มีฟอนต์หลายแบบตามที่แต่ละท่านได้คิดสร้างขึ้นมา ในบทนี้จะมีการกล่าวถึงโปรแกรมสร้างอักษรในหัวข้อ 10.8

2. มีหลายขนาด พร้อมทั้งสามารถลดหรือขยายขนาดได้อีกด้วย

3. แสดงอักษรได้หลายลักษณะ เช่น เอียงซ้าย เอียงขวา กลับหัว เรียงจากล่างขึ้นบน เรียงจากบนลงล่าง

4. แสดงอักษร ณ ตำแหน่งใดก็ได้ไม่จำกัดขอบเขตเป็นแถวและคอลัมน์

ตามปกติโปรแกรมคอมพิวเตอร์ที่ทำงานทางด้านกราฟิกส์ มักจะมีไฟล์อักษรซึ่งเรียกว่า **ฟอนต์ (font file)** อยู่ด้วยเสมอ ดังตัวอย่างในตารางที่ 10.1

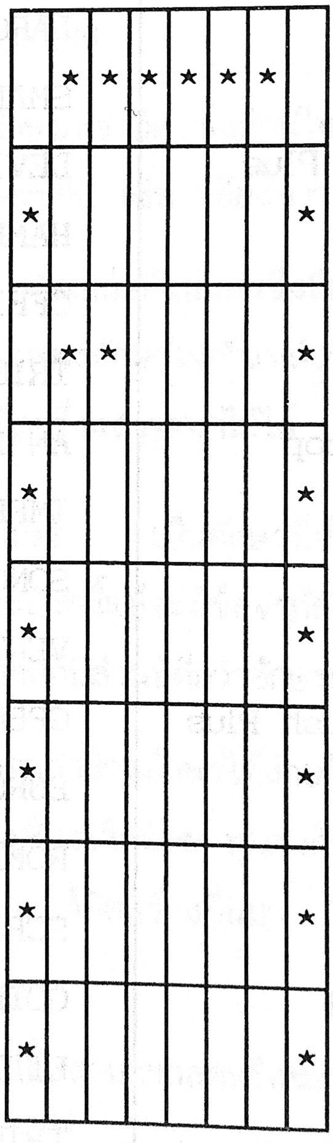
ตารางที่ 10.1 ตัวอย่างไฟล์อักษรที่มีในโปรแกรมกราฟิกส์บางโปรแกรม

ชื่อโปรแกรม	ตัวอย่างชื่อไฟล์อักษร
ShowPartner F/X	CHICAGO.FON
	KORINNA.FON
	LARGE.FON
	SMALLEST.FON
PrintMaster Plus	DEVILLE.FNT
	HAMPTON.FNT
	OFFICE.FNT
	TRIBUNE.FNT
New Print Shop	AMADOR.PNF
	IMPERIAL.PNF
	SONOMA.PNF
	VENTURA.PNF
PC Paint Brush Plus	GREEK.FNT
	EURO.FNT
	ROMAN.FNT
	SCRIPT.FNT
Turbo Pascal	GOTH.CHR
	LITT.CHR
	TRIP.CHR
	SANS.CHR

10.3 วิธีการสร้างอักษรซอฟต์แวร์

อักษรที่เก็บอยู่ในไฟล์สำหรับใช้ในโปรแกรมกราฟิกส์มีวิธีการสร้าง 2 แบบเช่นเดียวกับการสร้างภาพกราฟิกส์คือ แบบบิตแมพ (bit-mapped) และแบบเวกเตอร์ (vector) หรือสโตรก (stroked) ซึ่งมีรายละเอียดดังต่อไปนี้

10.3.1 อักษรแบบบิตแมพ สร้างขึ้นโดยการกำหนดพิกเซลไว้ในตำแหน่งต่างๆ ภายในตารางสี่เหลี่ยมขนาดหนึ่ง พิกเซลเหล่านี้ทำให้ตามองเห็นเป็นอักษร ดังแสดงในรูปที่ 10.1



รูปที่ 10.1 อักษร ก ซึ่งสร้างด้วยวิธีบิตแมพ โดยใช้ตารางขนาด 8×8

จะเห็นได้ว่า อักษรแบบนี้มีวิธีการสร้างคล้ายกับการกำหนดจุดลงบนกระดาษกราฟ ถ้ากำหนดจุดที่ตำแหน่งต่างๆ ได้เหมาะสมก็จะมองเห็นเป็นอักษร 1 ตัว

10.3.2 อักษรแบบเวกเตอร์ สร้างขึ้นโดยการใช้คำสั่งลากเส้นตรงเชื่อมต่อจุดต่างๆ ในทิศทางและตำแหน่งที่กำหนดเอาไว้ เช่น การสร้างอักษร A ที่ตำแหน่ง 0,0 บนจอภาพซึ่งกำหนดให้มุมล่างซ้ายเป็นตำแหน่ง 0,0 จะมีสเตตเมนต์ ดังรูปที่ 10.2

```
MoveTo (0,0);
LineTo (3,8);
LineTo (6,0);
MoveTo (1.2)
LineTo (5,2);
```

รูปที่ 10.2 สเตตเมนต์สร้างอักษร A ที่ตำแหน่ง 0,0 บนจอภาพ

สเตตเมนต์แรก หมายถึง ให้เลื่อนซีพีไปที่ตำแหน่ง 0,0 คือที่มุมล่างซ้ายของจอภาพ สเตตเมนต์ที่ 2 หมายถึงให้ลากเส้นตรงจากตำแหน่ง 0,0 ไปที่ตำแหน่ง 3,8 ซึ่งเป็นเส้นด้านซ้ายของอักษร A สเตตเมนต์ที่ 3 หมายถึงให้ลากเส้นตรงจากตำแหน่ง 3,8 ไปที่ตำแหน่ง 6,0 ซึ่งเป็นด้านขวาของอักษร A สเตตเมนต์ที่ 4 หมายถึงให้เลื่อนซีพีไปที่ตำแหน่ง 1,2 สเตตเมนต์สุดท้ายหมายถึงให้ลากเส้นตรงจากตำแหน่ง 1,2 ไปที่ตำแหน่ง 5,2 ซึ่งเป็นเส้นกลางของอักษร A

จากสเตตเมนต์เหล่านี้ เรานำมาเขียนเป็นโปรแกรมสำหรับสร้างอักษร A บนจอภาพได้ดังต่อไปนี้

ตัวอย่าง โปรแกรม 10.1 ใช้สำหรับแสดงอักษร A ซึ่งสร้างขึ้นแบบเวกเตอร์

โปรแกรม 10.1

```
PROGRAM DrawAFix;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
  TrueY := GetMaxY - Y
END;

PROCEDURE A;
BEGIN
   MoveTo(0,TrueY(0));
   LineTo(3,TrueY(8));
   LineTo(6,TrueY(0));
   MoveTo(1,TrueY(2));
   LineTo(5,TrueY(2));
END;

BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  A;
  ReadLn;
  CloseGraph
END.
```



รูปที่ 10.3 อักษร A ที่สร้างจากโปรแกรม 10.1

จากสเตตเมนต์ในรูปที่ 10.2 จะเห็นว่าได้มีการกำหนดตำแหน่งต่างๆ ไว้เป็นค่าคงที่ ซึ่งมีผลให้ตัวอักษรถูกสร้าง ณ ตำแหน่งเดิมทุกครั้ง เพื่อให้สามารถสร้างตัวอักษรที่ตำแหน่งต่างๆ ได้ตามต้องการ เราจำเป็นจะต้องปรับปรุงสเตตเมนต์ให้ลากเส้นแบบสัมพันธ์กับตำแหน่งของซีพีในขณะนั้น ดังแสดงเป็นสเตตเมนต์ในรูปที่ 10.4

```
MoveTo (X,Y);
LineRel (3,-8);
LineRel (3,8);
MoveRel (-5,-2);
LineRel (4,0);
```

รูปที่ 10.4 สเตตเมนต์สร้างอักษร A ที่ตำแหน่ง X, Y ใด ๆ บนจอภาพ

สเตตเมนต์แรกหมายถึงให้เลื่อนซีพีไปที่ตำแหน่ง X,Y โดย X,Y เป็นค่าที่เรากำหนดให้แก่โปรแกรม สเตตเมนต์ที่ 2 หมายถึงจากตำแหน่ง X,Y ให้ลากเส้นไปที่ตำแหน่งห่างไปทางขวา 3 พิกเซลและห่างไปข้างบน 8 พิกเซล สเตตเมนต์ที่ 3 หมายถึง จากตำแหน่งปัจจุบันให้ลากเส้นไปที่ตำแหน่งห่างไปทางขวา 3 พิกเซลและห่างไปข้างล่าง 8 พิกเซล สเตตเมนต์ที่ 4 หมายถึงจากตำแหน่งปัจจุบันให้เลื่อนซีพีไปที่ตำแหน่งห่างไปทางซ้าย 5 พิกเซลและห่างไปข้างบน 2 พิกเซล สเตตเมนต์สุดท้ายหมายถึงจากตำแหน่งปัจจุบันให้ลากเส้นขนานกับแกน X ไปที่ตำแหน่งห่างไปทางขวา 4 พิกเซล

จากสเตตเมนต์เหล่านี้ ทำให้เราสามารถแสดงอักษร A ณ ตำแหน่ง X,Y ต่างๆ บนจอภาพได้ตามต้องการ

ตัวอย่าง โปรแกรม 10.2 ใช้สำหรับแสดงอักษร A ซึ่งสร้างขึ้นแบบเวกเตอร์ โดยเราสามารถกำหนดให้แสดงอักษร ณ ตำแหน่งต่างๆบนจอภาพได้ตามต้องการ

โปรแกรม 10.2

```
PROGRAM DrawARelative;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;

PROCEDURE A(X,Y : Integer);
BEGIN
   MoveTo (X,TrueY(Y));
   LineRel(3,-8);
   LineRel(3, 8);
   MoveRel(-5,-2);
   LineRel(4,0);
END;

BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   A(300,150);
   ReadLn;
   CloseGraph
END.
```

10.4 เปรียบเทียบอักษรแบบบิตแมพและอักษรแบบเวกเตอร์

หัวข้อที่ใช้สำหรับเปรียบเทียบอักษรสองชนิดนี้คือ ความเร็วของการแสดงที่จอภาพและความสามารถในการขยายขนาด ซึ่งมีข้อสรุปดังนี้

1. อักษรแบบบิตแมพ สามารถแสดงที่จอภาพได้เร็วกว่าอักษรแบบเวกเตอร์ สมมติว่าเราต้องการแสดงอักษรแบบบิตแมพขนาด 8 x 8 หนึ่งตัว ก็เพียงแต่ใช้คำสั่งย้ายข้อมูลขนาด 8 ไบต์จากหน่วยความจำที่เก็บฟอนต์ไปยังหน่วยความจำของจอภาพ (video display buffer) ภาพอักษรนั้นก็จะปรากฏบนจอภาพทันที อักษรที่เครื่องคอมพิวเตอร์ใช้ในเท็กซ์โหมด(text mode) เป็นอักษรชนิดนี้ ส่วนอักษรแบบเวกเตอร์จะใช้เวลามากกว่าสำหรับการแสดงอักษรแต่ละตัว เนื่องจากคอมพิวเตอร์ต้องทำตามคำสั่งที่มีจำนวนมากกว่า

2. การขยายขนาดอักษรแบบบิตแมพจะทำได้ไม่มาก เพราะจะมีผลให้สัดส่วนและรูปร่างของอักษรแตกต่างไปจากเดิม เนื่องจากการขยายขนาดทำโดยวิธีการเพิ่มพิกเซลเข้าไปในแนวตั้งและ/หรือในแนวนอนเท่านั้น อักษรที่ขยายโตขึ้นจึงมองเห็นเป็นตารางสี่เหลี่ยมเรียงต่อกัน ทำให้มีลักษณะผิดเพี้ยนไปจากเดิม ซึ่งเราสามารถสังเกตได้จากการทดลองขยายอักษรด้วยโปรแกรมตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 10.3 แสดงอักษร ก ที่สร้างแบบบิตแมพ ซึ่งมีทั้งขนาดปกติและขนาดขยาย

โปรแกรม 10.3

```
PROGRAM ExpandBitLetter;
USES Graph;
CONST
   MaxLine      =  8;
   PixelInLine  =  8;
```

```
   PixelString : ARRAY[1..MaxLine] OF STRING[PixelInLine] = (
                        '  ******  ',
                        '*        *',
                        ' **      *',
                        '*        *',
                        '*        *',
                        '*        *',
                        '*        *',
                        '*        *');
VAR GraphDriver,GraphMode : Integer;

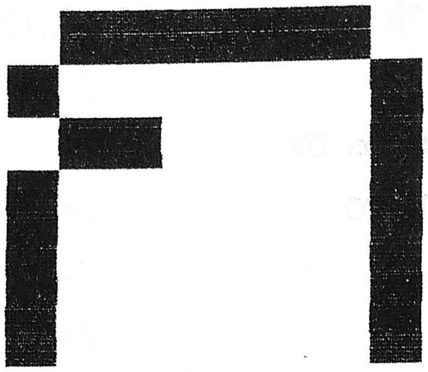
PROCEDURE DisplayBitLtt(X,Y : Integer);
VAR   ByteNo,ByteInLine,StartX,Line   : Integer;
BEGIN
   StartX := X;
   FOR Line := 1 TO MaxLine DO
   BEGIN
      ByteInLine := 1;
      FOR ByteNo := 1 TO PixelInLine DO
      BEGIN
         IF PixelString[Line,ByteNo] <> ' ' THEN PutPixel(X,Y,White);
         Inc(X);
      END;
      Inc(Y);   X := StartX;
   END;
END;

PROCEDURE ExpandBitLtt(X,Y,XScale,YScale : Integer);
VAR   ByteNo,ByteInLine,StartX,Line,Xs,Ys : Integer;
```

```
BEGIN
  StartX := X;
  FOR Line := 1 TO MaxLine DO
  FOR Ys := 1 TO YScale DO
  BEGIN
    ByteInLine := 1;
    FOR ByteNo := 1 TO PixelInLine DO
    FOR Xs      := 1 TO XScale DO
    BEGIN
      IF PixelString[Line,ByteNo] <> ' ' THEN PutPixel(X,Y,White);
      Inc(X);
    END;
    Inc(Y);  X := StartX;
  END;
END;

BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  DisplayBitLtt(100,100);        {normal size}
  ExpandBitLtt(200,100,20,14); {large size}
  ReadLn;
  CloseGraph
END.
```

ก



รูปที่ 10.5 ภาพที่ได้จากโปรแกรม 10.3

อักษรแบบเวกเตอร์จะขยายขนาดได้มากกว่า โดยสัดส่วนและรูปร่างของอักษรยังคงคล้ายเดิม ยิ่งกว่านั้นเรายังสามารถขยายเฉพาะความกว้างหรือความสูง เพื่อให้มองเห็นเป็นอักษรผอมหรืออ้วนกว่าปกติได้ด้วย ในงานคอมพิวเตอร์กราฟิกส์ที่ต้องการสร้างตัวอักษรขนาดใหญ่ซึ่งมีลักษณะต่างๆกัน เช่น ข้อความในป้ายประกาศ (banner) หรือชื่อเรื่อง (title) นิยมใช้อักษรชนิดนี้ เราจะสังเกตผลการทดลองขยายขนาดอักษรแบบเวกเตอร์ได้จากโปรแกรมตัวอย่างต่อไปนี้

ตัวอย่าง โปรแกรม 10.4 แสดงอักษร A ที่สร้างแบบเวกเตอร์ ซึ่งมีทั้งขนาดปกติและขนาดขยาย

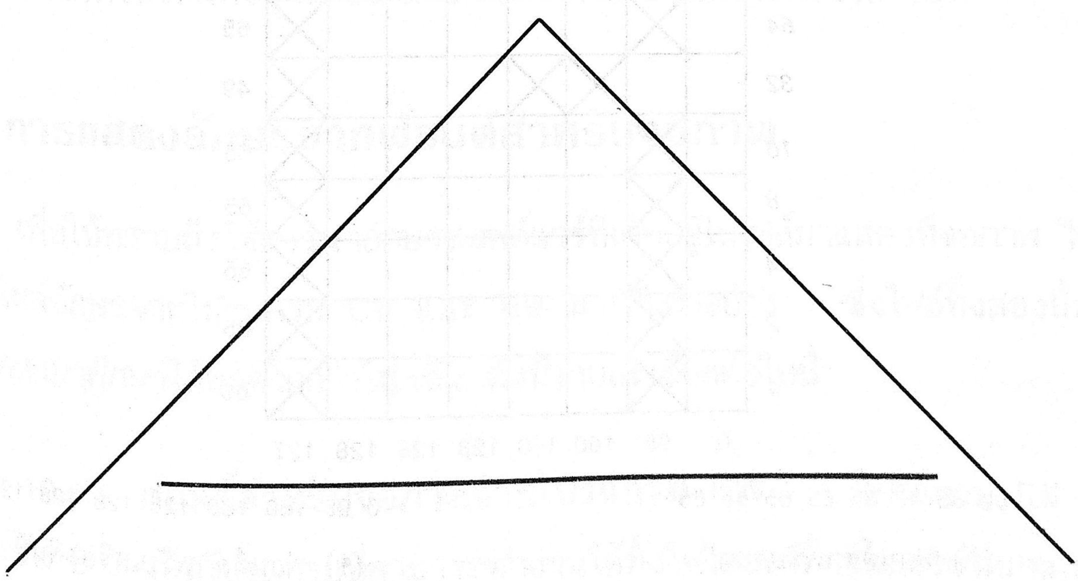
โปรแกรม 10.4

```
PROGRAM DrawAWithScale;
USES Graph;
VAR  GraphDriver,GraphMode : Integer;

FUNCTION TrueY(Y : Integer) : Integer;
BEGIN
   TrueY := GetMaxY - Y
END;
```

```
PROCEDURE A(X,Y,Wide,High : Integer);
BEGIN
  MoveTo (X,TrueY(Y));
  LineRel( 3*Wide,-8*High);
  LineRel( 3*Wide, 8*High);
  MoveRel(-5*Wide,-2*High);
  LineRel( 4*Wide, 0*High);
END;

BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver,GraphMode,'');
  A(50,100,1,1);       {normal size}
  A(100,100,75,20);    {large size}
  ReadLn;
  CloseGraph
END.
```



รูปที่ 10.6 ภาพที่ได้จากโปรแกรม 10.4

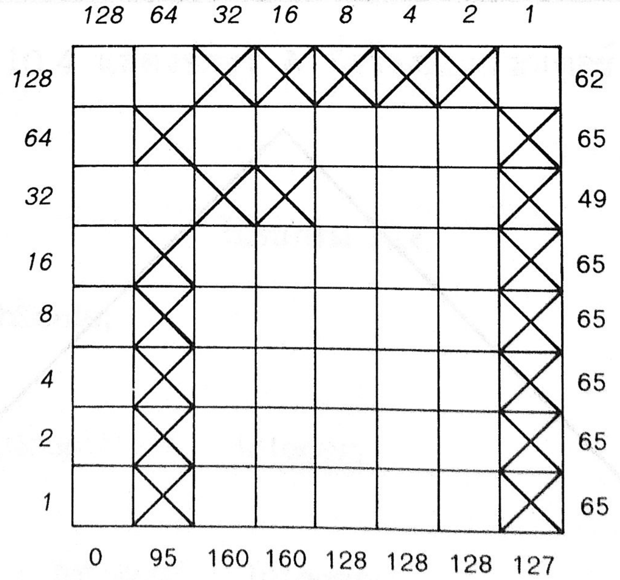
10.5 ฟอนต์สำหรับจอภาพและฟอนต์สำหรับเครื่องพิมพ์

เราจะมองเห็นอักษรจากฟอนต์ได้ โดยการสั่งให้คอมพิวเตอร์แสดงฟอนต์นั้นที่อุปกรณ์แสดงผลของคอมพิวเตอร์เช่น แสดงที่จอภาพ เครื่องพิมพ์และพล็อตเตอร์ เนื่องจากอุปกรณ์เหล่านี้มีการทำงานแตกต่างกัน ดังนั้น การสร้างฟอนต์จึงต้องมีวัตถุประสงค์ของการใช้ที่แน่นอน มิฉะนั้นจะใช้ไม่สะดวกหรืออาจจะใช้ไม่ได้เลย

สำหรับการใช้งานคอมพิวเตอร์ตามปกติ ส่วนมากมักจะสร้างฟอนต์ขึ้นโดยมีวัตถุประสงค์ที่จะนำมาใช้กับจอภาพและเครื่องพิมพ์ ดังนั้นในที่นี้จึงขอกล่าวถึงเฉพาะฟอนต์สำหรับอุปกรณ์ทั้งสองนี้

ทางด้านแบบของฟอนต์ ในที่นี้จะขอกล่าวถึงเฉพาะฟอนต์แบบบิตแมพเท่านั้น เนื่องจากเป็นฟอนต์ที่นิยมใช้กันมากในปัจจุบัน

ฟอนต์สำหรับจอภาพและฟอนต์สำหรับเครื่องพิมพ์มีความแตกต่างที่สำคัญในเรื่องลำดับการจัดเก็บข้อมูลของอักษรไว้ในไฟล์ โดยฟอนต์สำหรับจอภาพ จะเก็บข้อมูลอักษรแบบเรียงจากซ้ายไปขวา ส่วนฟอนต์สำหรับเครื่องพิมพ์จะเก็บข้อมูลอักษรแบบเรียงจากบนลงล่าง ดังแสดงในรูปที่ 10.7



62 65 46 65 65 65 65 65

(ก) ฟอนต์สำหรับจอภาพ

0 95 160 160 128 128 128 127

(ข) ฟอนต์สำหรับเครื่องพิมพ์

รูปที่ 10.7 การจัดลำดับข้อมูลของอักษรที่เก็บในไฟล์

โปรแกรมที่จำเป็นต้องใช้ฟอนต์บ่อยๆ ได้แก่ โปรแกรมประเภทเวิร์ดโพรเซสเซอร์ (word processer) ชนิดที่สามารถแสดงและพิมพ์อักษรได้หลายแบบ เช่น ตัวบาง ตัวหนา ตัวเอนและตัวขยาย ในที่นี้จะขอยกตัวอย่างโปรแกรมเวิร์ดโพรเซสเซอร์ภาษาไทย-อังกฤษ ซึ่งเป็นที่รู้จักกันแพร่หลาย ได้แก่ โปรแกรม CW และโปรแกรม RW

โปรแกรม RW มีชื่อเต็มว่า Rajavithi Word PC สร้างขึ้นโดย นายแพทย์ชุษณะ มะกรสาร ชมรมไมโครคอมพิวเตอร์ โรงพยาบาลราชวิถี เวอร์ชันล่าสุดขณะจัดทำหนังสือเล่มนี้คือ เวอร์ชัน 2.0 RW เวอร์ชันนี้สามารถทำงานได้ทั้งในโหมดเท็กซ์และโหมดกราฟิกส์ แตกต่างจากเวอร์ชันก่อนที่ทำงานในโหมดเท็กซ์เพียงอย่างเดียว ในดิสก์โปรแกรม RW มีฟอนต์สำหรับจอภาพรวม 2 ไฟล์คือ FONT1.CSL และ FONT2.CSL ซึ่งจะกล่าวถึงรายละเอียดในข้อ 10.6.1 และมีฟอนต์สำหรับเครื่องพิมพ์อีกหลายไฟล์ ซึ่งจะกล่าวถึงรายละเอียดในข้อ 10.7

โปรแกรม CW มีชื่อเต็มว่า CU Writer สร้างขึ้นโดย สถาบันบริการคอมพิวเตอร์ร่วมกับภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เวอร์ชันล่าสุดขณะจัดทำหนังสือเล่มนี้คือ เวอร์ชัน 1.52 CW ทุกเวอร์ชันทำงานในโหมดกราฟิกส์ ในดิสก์โปรแกรม CW มีฟอนต์สำหรับจอภาพรวม 2 ไฟล์ คือ NORMAL.FON และ ITALIC.FON ซึ่งจะกล่าวถึงรายละเอียดในข้อ 10.6.2 และมีฟอนต์สำหรับเครื่องพิมพ์อีกหลายไฟล์ ซึ่งจะกล่าวถึงรายละเอียดในข้อ 10.7

10.6 การแสดงอักษรจากฟอนต์สำหรับจอภาพ

เพื่อให้ทราบถึงวิธีการนำอักษรซอฟต์แวร์ที่เก็บอยู่ในไฟล์มาแสดงที่จอภาพ ในที่นี้จะใช้ไฟล์อักษรจากโปรแกรม CW และ RW มาเป็นตัวอย่าง ซึ่งไฟล์ทั้งสองนี้เราสามารถหามาศึกษาได้สะดวกกว่าไฟล์อื่น ดังมีรายละเอียดต่อไปนี้

10.6.1 ฟอนต์สำหรับจอภาพจากโปรแกรม RW โปรแกรม RW ตั้งแต่เวอร์ชัน 2.0 เป็นโปรแกรมที่สามารถทำงานได้ทั้งในโหมดเท็กซ์และในโหมดกราฟิกส์ ดังนั้นตัวอักษรที่ปรากฏบนจอภาพในโหมดกราฟิกส์จึงเป็นประเภทอักษรซอฟต์แวร์

1. ลักษณะฟอนต์ไฟล์ของโปรแกรม RW ไฟล์เก็บอักษรสำหรับแสดงที่จอภาพมี 2 ไฟล์คือ ไฟล์ FONT1.CSL ซึ่งเป็นอักษรตัวตรง และไฟล์ FONT2.CSL ซึ่งเป็นอักษรตัวเอน นอกจากความแตกต่างดังกล่าวนี้แล้ว อักษรในไฟล์ทั้งสองจะมีลักษณะเหมือนกันคือ

ก. เป็นอักษรแบบบิตแมพ

ข. ใช้ตารางขนาด 8 x 24 สำหรับอักษรแต่ละตัว

ค. อักษร 1 ตัวใช้หน่วยความจำ 24 ไบต์

ง. ในแต่ละฟอนต์ (หรือแต่ละไฟล์) ประกอบด้วยตัวอักษร 256 ตัวเรียงตามรหัส ASCII จาก 0 ถึง 255 โดยแบ่งเป็น 2 กลุ่มคือ

รหัส 0 – 127 อักษรเหมือนรหัส ASCII เดิม

รหัส 128 – 255 ปรับปรุงเป็นอักษรไทย

ดังนั้นไฟล์จึงมีขนาด 256 x 24 = 6144 ไบต์

ฉ. ข้อมูลในไฟล์เป็นข้อมูลประเภทรหัส ASCII

2. การนำอักษรจากไฟล์มาแสดงที่จอภาพมีขั้นตอนดังต่อไปนี้

ก. กำหนดจำนวนไบต์ของอักษร 1 ตัวคือ 24 ไบต์ให้เป็นขนาดของอาร์เรย์ดังนี้

```
TYPE
   CharByte  =  ARRAY [1..24] OF Byte
```

ข. เนื่องจากข้อมูลในไฟล์มีจำนวน 6144 ไบต์ เก็บเรียงต่อกันตามลำดับ จึงต้องแบ่งข้อมูลออกเป็นชุดรวม 256 ชุด ชุดละ 24 ไบต์ ซึ่งเท่ากับข้อมูลของอักษร 1 ตัว วิธีแบ่งข้อมูลทำได้โดยการกำหนดชนิดของข้อมูลให้เป็นเรคอร์ดของอาร์เรย์ดังนี้

```
TYPE
   RecordOfchar  =  RECORD
```

```
            OneChar : CharByte
        END;
VAR
    CharRec  :  RecordOfChar
```

ค. เพื่อความสะดวกต่อการอ้างถึงอักษรแต่ละตัว จึงกำหนดอาร์เรย์สำหรับเก็บข้อมูลของอักษรแต่ละตัวเรียงตามรหัส ASCII ดังนี้

```
VAR
    Ascii   :   ARRAY [0..255] Of CharByte
```

ง. เปิดไฟล์ FONT1.CSL หรือ FONT2.CSL เพื่ออ่านข้อมูลครั้งละ 24 ไบต์เข้ามาในฟิลด์ OneChar แล้วเก็บไว้ในอาร์เรย์ Ascii เรียงตามลำดับจาก 0 ถึง 255 เมื่ออ่านจบไฟล์ข้อมูลอักษรทุกตัวจากไฟล์จะถูกเก็บอยู่ใน Ascii ในลักษณะเรียงลำดับตามรหัส ASCII นั่นคือ Ascii[65] เป็นอักษร A Ascii[161] เป็นอักษร ก เป็นต้น

จ. แสดงอักษร 1 ตัวโดยวิธีการลากเส้นขนาด 1 ไบต์จำนวน 24 เส้นเรียงตามลำดับลงมาข้างล่างด้วยโพรซีเยอร์ Line ลักษณะของเส้นแต่ละเส้นเป็นข้อมูลจากไบต์ที่ 1 ถึงไบต์ที่ 24 ซึ่งกำหนดด้วยโพรซีเยอร์ SetLineStyle โดยระบุพารามิเตอร์ตัวแรก (ทำหน้าที่บอกลักษณะของเส้น) ให้เป็น UserBitLn หรือ 4 ซึ่งหมายถึงเส้นมีลักษณะตามที่ผู้เขียนโปรแกรมกำหนด

การอ้างถึงตัวอักษรที่จะนำมาแสดง ทำได้โดยการกำหนดค่าอินเด็กซ์(index) ของ Ascii ดังนี้

Ascii [รหัสอักษร, ไบต์ที่ 1 ถึง 24]

เช่น Ascii [65, ByteNo] หมายถึง ให้แสดงอักษร A

เนื่องจาก ByteNo มี 24 ไบต์ จึงต้องใช้ลูป FOR ลากเส้นรวม 24 รอบ

ตัวอย่าง โปรแกรม 10.5 ใช้สำหรับแสดงอักษรจากไฟล์ FONT1.CSL และ FONT2.CSL ซึ่งมีอยู่ในดิสก์โปรแกรม RW เมื่อรันโปรแกรมเราจะต้องป้อนชื่อไฟล์ใดไฟล์หนึ่งจากสองไฟล์นี้ ถ้าจำเป็นอาจจะต้องระบุชื่อดิสก์ไดรฟ์ที่มีไฟล์นั้นอยู่ด้วยก็ได้ เช่น B:FONT1.CSL โปรแกรมจะอ่านไฟล์ และนำฟอนต์ชุดนั้นมาแสดงที่จอภาพทุกอักษรตั้งแต่รหัส 0 – 255 พารามิเตอร์ตัวที่ 3 ของ SetLineStyle ทำหน้าที่กำหนดความหนาของเส้น ซึ่งเลือกระบุได้ 2 ค่า คือ 1 = เส้นบาง และ 3 = เส้นหนา การเปลี่ยนพารามิเตอร์ตัวนี้จะมีผลให้เราเห็นอักษรเป็นตัวบางหรือตัวหนาแตกต่างกัน

โปรแกรม 10.5

```
PROGRAM FontRWCs6144;
USES Graph;
TYPE CharByte        = ARRAY[1..24] OF Byte;
     RecordOfChar  = RECORD
                       OneChar : CharByte;
                     END;
VAR GraphDriver,GraphMode : Integer;
    FontFile   : File OF RecordOfChar;
    FileName   : STRING[14];
    CharRec    : RecordOfChar;
    Ascii      : ARRAY [0..255] OF CharByte;
    Row,Col    : Integer;
    CharNo     : Byte;

PROCEDURE ReadFont;
VAR CharNo : Byte;
BEGIN
  Assign(FontFile,FileName);
```

```
  Reset(FontFile);
  CharNo := 0;
  WHILE NOT Eof(FontFile) DO
  BEGIN
    Read(FontFile,CharRec);
    Ascii[CharNo] := CharRec.OneChar;
    Inc(CharNo);
  END;
  Close(FontFile);
END;

PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
BEGIN
 FOR ByteNo :=1 TO 24 DO
 BEGIN
   SetLineStyle(UserBitLn,Ascii[CharNo,ByteNo],1);
   IF GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
   Line(X,Y, X+15,Y);
   Inc(Y,1);
 END;
END;

BEGIN
  WriteLn('DISPLAY FONT FROM RW 2.0');
  Write('Enter FONT1.CSL or FONT2.CSL : ');
  ReadLn(FileName);
  ReadFont;
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  ClearDevice;
```

```
  Row    := 30;
  Col    := 1;
  CharNo := 0;
  REPEAT
     DisplayChar(Col,Row,CharNo);
     Inc(CharNo);
     Inc(Col,10);
     IF Col > GetMaxX - 10 THEN
     BEGIN
        Col := 1;
        Inc(Row,30)
     END;
  UNTIL CharNo = 255;
  ReadLn;
  CloseGraph
END.
```

10.6.2 ฟอนต์สำหรับจอภาพจากโปรแกรม CW เนื่องจาก CW เป็นโปรแกรมที่ทำงานในโหมดกราฟิกส์ ดังนั้นตัวอักษรที่ปรากฏบนจอภาพจึงเป็นประเภทอักษรซอฟต์แวร์

1. ลักษณะฟอนต์ไฟล์ของโปรแกรม CW ไฟล์เก็บอักษรสำหรับแสดงที่จอภาพมี 2 ไฟล์คือ ไฟล์ NORMAL.FON ซึ่งเป็นอักษรตัวตรง และไฟล์ ITALIC.FON ซึ่งเป็นอักษรตัวเอน นอกจากความแตกต่างดังกล่าวนี้แล้ว อักษรในไฟล์ทั้งสองจะมีลักษณะเหมือนกัน คือ

ก. เป็นอักษรแบบบิตแมพ

ข. ใช้ตารางขนาด 8 x 20 สำหรับอักษรแต่ละตัว

ค. อักษร 1 ตัวใช้หน่วยความจำ 20 ไบต์

ง. ในแต่ละฟอนต์ (หรือแต่ละไฟล์) ประกอบด้วยตัวอักษร 256 ตัวเรียงตามรหัส ASCII จาก 0 ถึง 255 โดยแบ่งเป็น 2 กลุ่มคือ

รหัส 0 - 127 อักษรเหมือนรหัส ASCII เดิม

รหัส 128 - 255 ปรับปรุงเป็นอักษรไทย

ดังนั้นไฟล์จึงมีขนาด 256 x 20 = 5120 ไบต์

จ. ข้อมูลในไฟล์เป็นข้อมูลประเภทรหัส ASCII

2. การนำอักษรจากไฟล์มาแสดงที่จอภาพมีขั้นตอนดังต่อไปนี้

ก. กำหนดจำนวนไบต์ของอักษร 1 ตัวคือ 20 ไบต์ให้เป็นขนาดของอาร์เรย์ดังนี้

```
TYPE
   CharByte   =   ARRAY [1..20] OF Byte
```

ข. เนื่องจากข้อมูลในไฟล์มีจำนวน 5120 ไบต์ เก็บเรียงต่อกันตามลำดับ จึงต้องแบ่งข้อมูลออกเป็นชุดรวม 256 ชุด ชุดละ 20 ไบต์ ซึ่งเท่ากับข้อมูลของอักษร 1 ตัว วิธีแบ่งข้อมูลทำได้โดยการกำหนดชนิดของข้อมูลให้เป็นเรคอร์ดของอาร์เรย์ดังนี้

```
TYPE
   RecordOfchar   =   RECORD
                          OneChar : CharByte
                      END;
VAR
   CharRec   :   RecordOfChar
```

ค. เพื่อความสะดวกต่อการอ้างถึงอักษรแต่ละตัว จึงกำหนดอาร์เรย์สำหรับเก็บข้อมูลของอักษรแต่ละตัวเรียงตามรหัส ASCII ดังนี้

```
VAR
   Ascii  :  ARRAY [0..255] Of CharByte
```

ง. เปิดไฟล์ NORMAL.FON หรือ ITALIC.FON เพื่ออ่านข้อมูลครั้งละ 20 ไบต์เข้ามาในฟิลด์ OneChar แล้วเก็บไว้ในอาร์เรย์ Ascii เรียงตามลำดับจาก 0 ถึง 255 เมื่ออ่านจบไฟล์ข้อมูลอักษรทุกตัวจากไฟล์จะถูกเก็บอยู่ใน Ascii ในลักษณะเรียงลำดับตามรหัส ASCII นั่นคือ Ascii[65] เป็นอักษร A Ascii[161] เป็นอักษร ก เป็นต้น

จ. แสดงอักษร 1 ตัว โดยวิธีการลากเส้นขนาด 1 ไบต์จำนวน 20 เส้น เรียงตามลำดับลงมาข้างล่าง ด้วยโพรซีเยอร์ Line ลักษณะของเส้นแต่ละเส้นเป็นข้อมูลจากไบต์ที่ 1 ถึงไบต์ที่ 20 ซึ่งกำหนดด้วยโพรซีเยอร์ SetLineStyle โดยระบุพารามิเตอร์ตัวแรก ให้เป็น UserBitLn หรือ 4 ซึ่งหมายถึงเส้นมีลักษณะตามที่ผู้เขียนโปรแกรมกำหนด

การอ้างถึงตัวอักษรที่จะนำมาแสดง ทำได้โดยการกำหนดค่าอินเด็กซ์(index) ของ Ascii ดังนี้

Ascii [รหัสอักษร, ไบต์ที่ 1 ถึง 20]

เช่น Ascii [65, ByteNo] หมายถึง ให้แสดงอักษร A

เนื่องจาก ByteNo มี 20 ไบต์ จึงต้องใช้ลูป FOR ลากเส้นรวม 20 รอบ

ตัวอย่าง โปรแกรม 10.6 ใช้สำหรับแสดงอักษรจากไฟล์ NORMAL.FON และ ITALIC.FON ซึ่งมีอยู่ในดิสก์โปรแกรม CW เมื่อรันโปรแกรมเราจะต้องป้อนชื่อไฟล์ใดไฟล์หนึ่งจากสองไฟล์นี้ โปรแกรมจะอ่านไฟล์ และนำฟอนต์ชุดนั้นมาแสดงที่จอภาพทุกอักษรตั้งแต่รหัส 0 ถึง 255 การทำให้เห็นอักษรเป็นตัวบางหรือตัวหนา ทำได้โดยการเปลี่ยนพารามิเตอร์ตัวที่ 3 ของโพรซีเยอร์ SetLineStyle ซึ่งกำหนดได้ 2 ค่า คือ 1 = เส้นบาง และ 3 = เส้นหนา

โปรแกรม 10.6

```
PROGRAM FontCWSc5120;
USES Graph;
TYPE CharByte          = ARRAY[1..20] OF Byte;
     RecordOfChar  = RECORD
                          OneChar : CharByte;
                      END;
VAR GraphDriver,GraphMode : Integer;
    FontFile   : File OF RecordOfChar;
    FileName   : STRING[14];
    CharRec    : RecordOfChar;
    Ascii      : ARRAY [0..255] OF CharByte;
    Row,Col    : Integer;
    CharNo     : Byte;

PROCEDURE ReadFont;
VAR CharNo : Byte;
BEGIN
   Assign(FontFile,FileName);
   Reset(FontFile);
   CharNo := 0;
   WHILE NOT Eof(FontFile) DO
   BEGIN
      Read(FontFile,CharRec);
      Ascii[CharNo] := CharRec.OneChar;
      Inc(CharNo);
   END;
   Close(FontFile);
END;
```

```
PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
BEGIN
  FOR ByteNo :=1 TO 20 DO
  BEGIN
     SetLineStyle(UserBitLn,Ascii[CharNo,ByteNo],1);
     IF GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
     Line(X,Y, X+15,Y);
     Inc(Y,1);
  END;
END;

BEGIN
   WriteLn('DISPLAY FONT FROM CW');
   Write('Enter NORMAL.FON or ITALIC.FON : ');
   ReadLn(FileName);
   ReadFont;
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   ClearDevice;
   Row     := 30;
   Col     := 1;
   CharNo  := 0;
   REPEAT
     DisplayChar(Col,Row,CharNo);
     Inc(CharNo);
     Inc(Col,10);
     IF Col > GetMaxX - 10 THEN
     BEGIN
        Col := 1;
```

```
      Inc(Row,30)
    END;
  UNTIL CharNo = 255;
  ReadLn;
  CloseGraph
END.
```

10.7 การแสดงอักษรจากฟอนต์สำหรับเครื่องพิมพ์

แม้ว่าฟอนต์สำหรับเครื่องพิมพ์จะถูกสร้างไว้สำหรับงานพิมพ์โดยเฉพาะ แต่เราก็สามารถนำมาแสดงที่จอภาพได้ ด้วยวิธีการที่จะกล่าวถึงในหัวข้อนี้ เพื่อความสะดวกแก่การศึกษา เราจะนำไฟล์อักษรสำหรับเครื่องพิมพ์จากโปรแกรม RW และ CW มาเป็นตัวอย่าง โดยเลือกศึกษาเฉพาะฟอนต์สำหรับเครื่องพิมพ์ชนิดดอตเมตริกซ์ (dot matrix)

10.7.1 ลักษณะและส่วนประกอบของฟอนต์สำหรับเครื่องพิมพ์ โปรแกรม RW มีฟอนต์สำหรับเครื่องพิมพ์ชนิดดอตเมตริกซ์รวม 3 กลุ่มโดยแบ่งกลุ่มให้ทราบด้วยชนิดของไฟล์ ซึ่งมีไฟล์ 3 ชนิดคือ LQ LQ1 และ NLQ ในดิสก์โปรแกรม RW จะมีโปรแกรมสำหรับสร้าง แก้ไข หรือเพิ่มเติมอักษรในไฟล์อยู่ด้วยรวม 3 โปรแกรม เพื่อใช้ดำเนินการกับไฟล์ในกลุ่มต่างๆ ดังนี้

ชื่อโปรแกรม	**ดำเนินการกับไฟล์ชนิด**
FONTGEN.COM	NLQ
FONTLQ1.EXE	LQ1
FONTLQ.EXE	LQ

ไฟล์อักษรสำหรับเครื่องพิมพ์ของโปรแกรม RW แต่ละไฟล์มีลักษณะและส่วนประกอบต่างๆ ดังแสดงในตารางที่ 10.2

ตารางที่ 10.2 ลักษณะและส่วนประกอบต่าง ๆ ของไฟล์อักษรในโปรแกรม RW

ชื่อ ชนิดไฟล์	ขนาดไฟล์	ขนาดตาราง	ไบต์ต่ออักษร	ลักษณะอักษร
FONT1.LQ	12096	24 x 18	54	กลาง, ตรง
FONT2.LQ	12096	24 x 18	54	กลาง, ตรง
FONTS.LQ	8064	16 x 18	36	เล็ก, ตรง
FONT1.LQ1	24192	24 x 36	108	ใหญ่, ตรง
FONT2.LQ1	24192	24 x 36	108	ใหญ่, เอียง
FONTS.LQ1	16128	16 x 36	72	กลาง, ตรง
FONT1.NLQ	9856	16 x 22	44	เล็ก, ตรง
FONT2.NLQ	9856	16 x 22	44	เล็ก, เอียง
FONTS.NLQ	9856	16 x 22	44	จิ๋ว , ตรง
FONTA1.NLQ	8064	16 x 18	36	เล็ก, ตรง
FONTA2.NLQ	8064	16 x 18	36	เล็ก, เอียง

โปรแกรม CW มีฟอนต์สำหรับเครื่องพิมพ์ชนิดดอตเมตริกซ์รวม 2 กลุ่ม โดยแบ่งกลุ่มด้วยชนิดของไฟล์คือ ไฟล์ชนิด PRN และ P24 ซึ่งแต่ละไฟล์มีลักษณะและส่วนประกอบดังแสดงในตารางที่ 10.3

ตัวอย่างที่ 10.3 ลักษณะและส่วนประกอบต่าง ๆ ของไฟล์อักษรในโปรแกรม CW

ชื่อ ชนิดไฟล์	ขนาดไฟล์	ขนาดตาราง	ไบต์ต่ออักษร	ลักษณะอักษร
NORMAL.P24	12096	24 x 18	54	กลาง,ตรง
NORMALS.P24	8064	16 x 18	36	เล็ก,ตรง
ITALIC.P24	12096	24 x 18	54	กลาง,เอียง
ITALICS.P24	8064	16 x 18	36	เล็ก,ตรง
NORMAL.PRN	11264	16 x 22	44	เล็ก,ตรง
NORMALS.PRN	11264	16 x 22	44	จิ๋ว,ตรง
ITALIC.PRN	11264	16 x 22	44	เล็ก,เอียง

จากตารางทั้งสองนี้ จะเห็นได้ว่ามีทั้งส่วนที่เหมือนกันและต่างกันปรากฏอยู่ อย่างไรก็ตาม เราสามารถสรุปลักษณะและส่วนประกอบรวมๆ ของไฟล์อักษรเหล่านี้ได้ดังต่อไปนี้

1. ข้อมูลในไฟล์มีชนิดเป็นรหัส ASCII
2. อักษรในไฟล์เป็นอักษรแบบบิตแมพ
3. ในแต่ละฟอนต์ (หรือแต่ละไฟล์) ประกอบด้วยตัวอักษร 224 ตัวเรียงตามตามรหัส ASCII จาก 32 ถึง 255 โดยแบ่งเป็น 2 กลุ่มคือ

รหัส 32 - 127 อักษรเหมือนรหัส ASCII เดิม

รหัส 128 - 255 อักษรไทย

ฟอนต์ที่มี 256 ตัวอักษรได้แก่ฟอนต์ในโปรแกรม CW ซึ่งมีชนิดของไฟล์เป็น PRN ประกอบด้วย 3 ไฟล์คือ NORMAL.PRN, NORMALS.PRN และ ITALIC.PRN ส่วนที่ทำให้มากกว่าไฟล์อื่นก็คือรหัส ASCII จาก 0 ถึง 31 ซึ่งเป็นอักษรว่างทั้งหมด

4. ขนาดตารางสำหรับอักษร 1 ตัว แบ่งเป็น 5 ขนาดคือ

ก. ขนาด 24 x 36

ข. ขนาด 24 x 18

ค. ขนาด 16 x 36

ง. ขนาด 16 x 22

จ. ขนาด 16 x 18

5. หน่วยความจำสำหรับอักษร 1 ตัวมี 4 ขนาดคือ 108 54 72 44 และ 36 ไบต์

10.7.2 โปรแกรมแสดงอักษรจากฟอนต์สำหรับเครื่องพิมพ์ เนื่องจากขนาดตารางของตัวอักษรเป็นค่าที่ทำให้โปรแกรมแสดงอักษรมีความแตกต่างกัน ดังนั้นเพื่อเป็นตัวอย่างในการศึกษาจึงควรมีโปรแกรม 4 โปรแกรมสำหรับใช้แสดงอักษรขนาดต่างๆ คือ 24 x 36, 24 x 18, 16 x 22 และ 16 x 18 ซึ่งหมายความว่าโปรแกรมดังกล่าวนี้ สามารถใช้ได้กับฟอนต์ของทั้งโปรแกรม CW และโปรแกรม RW นอกจากนั้นโปรแกรมยังใช้กับไฟล์อักษรจากโปรแกรมอื่นที่มีลักษณะตรงกันได้ด้วย

เพื่อความสะดวกต่อการป้อนชื่อไฟล์ให้แก่โปรแกรม ในที่นี้จึงตั้งชื่อโปรแกรมตามขนาดของไฟล์คือ โปรแกรม FON8064,FON11264,FON12096 และ FON24192 ด้วยเหตุผลที่ว่าขนาดของไฟล์เป็นสิ่งที่เราทราบได้ง่ายโดยการใช้คำสั่ง DIR ของ DOS และขนาดของไฟล์จะมีความสัมพันธ์กับขนาดของตัวอักษร ซึ่งจะสังเกตได้จากตารางที่ 10.2 และ 10.3

วิธีการนำอักษรจากไฟล์มาแสดงที่จอภาพของทุกโปรแกรมมีขั้นตอนคล้ายกัน ในที่นี้จะใช้ไฟล์ขนาด 12096 ไบต์ ซึ่งเก็บอักษรขนาด 24 x 18 พิกเซล มาเป็นตัวอย่างประกอบคำอธิบาย เนื่องจากมีไฟล์ขนาดนี้ทั้งใน RW และ CW คือ

ใน RW มีไฟล์ FONT1.LQ
FONT2.LQ

ใน CW มีไฟล์ NORMAL.P24
ITALIC.P24

การนำอักษรจากไฟล์สำหรับเครื่องพิมพ์มาแสดงที่จอภาพมีขั้นตอนดังต่อไปนี้

1. กำหนดจำนวนไบต์ของอักษร 1 ตัวคือ 54 ไบต์ ให้เป็นขนาดของอาเรย์ดังนี้

```
TYPE
    CharByte   =   ARRAY [1..54] OF Byte
```

2. เนื่องจากข้อมูลในไฟล์มีจำนวน 12096 ไบต์เก็บเรียงต่อกันตามลำดับ จึงต้องแบ่งข้อมูลออกเป็นชุด รวม 224 ชุด ชุดละ 54 ไบต์ ซึ่งเท่ากับข้อมูลของอักษร 1 ตัว วิธีแบ่งข้อมูลทำได้โดยการกำหนดชนิดของข้อมูลให้เป็นเรคอร์ดของอาร์เรย์ดังนี้

```
TYPE
    RecordOfChar   =   RECORD
                          OneChar : CharByte
                       END;
VAR
    CharRec   :   RecordOfChar
```

3. เพื่อความสะดวกต่อการอ้างถึงอักษรแต่ละตัว จึงกำหนดอาร์เรย์สำหรับเก็บข้อมูลของอักษรแต่ละตัวเรียงตามรหัส ASCII ดังนี้

```
VAR
    Ascii   :   ARRAY [0..255] Of CharByte
```

4. เปิดไฟล์ FONT1.LQ, FONT2.LQ, NORMAL.P24 หรือ ITALIC.P24 เพื่ออ่านข้อมูลครั้งละ 54 ไบต์เข้าในฟิลด์ OneChar แล้วเก็บไว้ในอาร์เรย์ Ascii เรียงตามลำดับจาก 32 ถึง 255 เมื่ออ่านจบไฟล์ ข้อมูลอักษรทุกตัวจากไฟล์จะถูกเก็บอยู่ใน Ascii ในลักษณะเรียงลำดับตามรหัส ASCII นั่นคือ Ascii[65] เป็นอักษร A Ascii[161] เป็นอักษร ก เป็นต้น

5. แสดงอักษร 1 ตัวโดยวิธีการลากเส้นขนาด 3 ไบต์จำนวน 18 เส้น การลากเส้นแต่ละเส้นจะลากจากด้านล่างขึ้นข้างบน ด้วยสเตตเมนต์ต่อไปนี้

```
Line(X, Y, X, Y-15)
```

และให้เส้นซึ่งได้จากข้อมูลแต่ละไบต์เรียงต่อกันตามลำดับจนครบ 3 ไบต์ การทดสอบครบจำนวนทุกๆ 3 ไบต์ ใช้สเตตเมนต์ต่อไปนี้

IF ByteNo MOD 3 = 0 THEN (ทดสอบว่า 3 หารลงตัว)

เมื่อลากเส้นครบ 3 ไบต์ จะได้เส้นในแนวตั้ง 1 เส้น ต่อจากนั้นจึงเริ่มลากเส้นที่ 2 ณ จุดเริ่มต้นจุดเดียวกันกับเส้นแรก และให้อยู่ถัดไปทางขวาของเส้นแรกทำเช่นนี้ไปตามลำดับจนครบ 18 เส้น เราจะเห็นอักษร 1 ตัวบนจอภาพ

ลักษณะเส้นของข้อมูลแต่ละไบต์กำหนดด้วยโพรซีเยอร์ SetLineStyle ตามวิธีที่เราได้ศึกษามาแล้ว

การอ้างถึงตัวอักษรที่จะนำมาแสดง ทำได้โดยการกำหนดค่าอินเด็กซ์ของ Ascii ดังนี้

Ascii [รหัสอักษร, ไบต์ที่ 1 ถึง 54]
เช่น Ascii [65, ByteNo] หมายถึง ให้แสดงอักษร A

เนื่องจาก ByteNo มี 54 ไบต์ จึงต้องใช้ลูป FOR ลากเส้นรวม 54 รอบ

ตัวอย่าง โปรแกรม 10.7 ใช้สำหรับแสดงอักษรบนจอภาพ โดยนำอักษรสำหรับเครื่องพิมพ์ทุกตัวที่มีอยู่ในไฟล์ขนาด 12096 ไบต์ มาแสดงตามลำดับรหัส ASCII ตั้งแต่รหัส 32 ถึง 255 ไฟล์ที่ใช้ได้กับโปรแกรมนี้คือ FONT1.LQ, FONT2.LQ, NORMAL.P24 และ ITALIC.P24

โปรแกรม 10.7

```
PROGRAM DisplayFont12096Byte;
USES Graph;
TYPE CharByte        = ARRAY[1..54] OF Byte;
   RecordOfChar  = RECORD
                    OneChar : CharByte;
                  END;
VAR GraphDriver,GraphMode : Integer;
   FontFile   : File OF RecordOfChar;
   FileName   : STRING[14];
   CharRec    : RecordOfChar;
   Ascii      : ARRAY [32..255] OF CharByte;
   Row,Col    : Integer;
   CharNo     : Byte;

PROCEDURE ReadFont;
VAR CharNo : Byte;
BEGIN
  Assign(FontFile,FileName);
  Reset(FontFile);
  CharNo := 32;
  WHILE NOT Eof(FontFile) DO
  BEGIN
    Read(FontFile,CharRec);
    Ascii[CharNo] := CharRec.OneChar;
    Inc(CharNo);
  END;
  Close(FontFile);
END;
```

```
PROCEDURE SwapBit(VAR NewBit : Byte);
VAR I,X,Y,Z : Integer;
BEGIN
   Z := 1; Y := NewBit; NewBit := 0;
   FOR I := 1 TO 8 DO
   BEGIN
      X := Y DIV (128 DIV Z);
      X := X * Y;
      Y := Y MOD (128 DIV Z);
      Z := Z * 2;
      NewBit := NewBit + X
   END
END;
PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
    StartY : Integer;
BEGIN
  StartY := Y;
  FOR ByteNo :=1 TO 54 DO
  BEGIN
     IF GrDriver <> 7 THEN SwapBit(Ascii[CharNo,ByteNo]);
     SetLineStyle(UserBitLn,Ascii[CharNo,ByteNo],1);
     Line(X,Y, X,Y-15);
     Inc(Y,8);
     IF ByteNo MOD 3 = 0 THEN
     BEGIN
        Y := StartY;
        Inc(X)
     END;
```

```
 END;
END;

BEGIN
  WriteLn('DISPLAY FONT FROM FILE 12096 BYTE');
  Write('Enter FONT1.LQ, FONT2.LQ, NORMAL.P24 or ITALIC.P24 : ');
  ReadLn(FileName);
  ReadFont;
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  ClearDevice;
  Row     := 30;
  Col     := 1;
  CharNo := 32;
  REPEAT
     DisplayChar(Col,Row,CharNo);
     Inc(CharNo);
     Inc(Col,30);
     IF Col > GetMaxX - 30 THEN
     BEGIN
        Col := 1;
        Inc(Row,30)
     END;
  UNTIL CharNo = 255;
  ReadLn;
  CloseGraph
END.
```



รูปที่ 10.8 อักษรจากไฟล์ FONT1.LQ

ตัวอย่าง โปรแกรม 10.8 ใช้สำหรับแสดงอักษรบนจอภาพ โดยนำอักษรสำหรับเครื่องพิมพ์ทุกตัวที่มีอยู่ในไฟล์ขนาด 24192 ไบต์ มาแสดงตามลำดับรหัส ASCII ตั้งแต่รหัส 32 ถึง 255 ไฟล์ที่ใช้ได้กับโปรแกรมนี้คือ FONT1.LQ1 และ FONT2.LQ1

โปรแกรม 10.8

```
PROGRAM DisplayFont24192Byte;
USES Graph;
TYPE CharByte         = ARRAY[1..108] OF Byte;
     RecordOfChar  = RECORD
                        OneChar : CharByte;
                     END;
```

```
VAR GraphDriver,GraphMode : Integer;
    FontFile   : File OF RecordOfChar;
    FileName   : STRING[14];
    CharRec    : RecordOfChar;
    Ascii      : ARRAY [32..255] OF CharByte;
    Row,Col    : Integer;
    CharNo     : Byte;

PROCEDURE ReadFont;
VAR CharNo : Byte;
BEGIN
   Assign(FontFile,FileName);
   Reset(FontFile);
   CharNo := 32;
   WHILE NOT Eof(FontFile) DO
   BEGIN
      Read(FontFile,CharRec);
      Ascii[CharNo] := CharRec.OneChar;
      Inc(CharNo);
   END;
   Close(FontFile);
END;
PROCEDURE SwapBit(VAR NewBit : Byte);
VAR I,X,Y,Z : Integer;
BEGIN
   Z := 1; Y := NewBit; NewBit := 0;
   FOR I := 1 TO 8 DO
   BEGIN
      X := Y DIV (128 DIV Z);
      X := X * Y;
```

```
      Y := Y MOD (128 DIV Z);
      Z := Z * 2;
      NewBit := NewBit + X
   END
END;

PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
    StartY : Integer;
BEGIN
  StartY := Y;
  FOR ByteNo :=1 TO 108 DO
  BEGIN
     IF GrDriver <> 7 THEN SwapBit(Ascii[CharNo,ByteNo]);
     SetLineStyle(UserBitLn,Ascii[CharNo,ByteNo],1);
     Line(X,Y, X,Y-15);
     Inc(Y,8);
     IF ByteNo MOD 3 = 0 THEN
     BEGIN
        Y := StartY;
        Inc(X)
     END;
  END;
END;

BEGIN
   WriteLn('DISPLAY FONT FROM RW');
   Write('Enter FONT1.LQ1 or FONT2.LQ1 : ');
   ReadLn(FileName);
```

```
  ReadFont;
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  ClearDevice;
  Row    := 30;
  Col    := 1;
  CharNo := 32;
  REPEAT
     DisplayChar(Col,Row,CharNo);
     Inc(CharNo);
     Inc(Col,30);
     IF Col > GetMaxX - 30 THEN
     BEGIN
        Col := 1;
        Inc(Row,30)
     END;
  UNTIL CharNo = 255;
  ReadLn;
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 10.9 ใช้สำหรับแสดงอักษรบนจอภาพ โดยนำอักษรสำหรับเครื่องพิมพ์ทุกตัวที่มีอยู่ในไฟล์ขนาด 8064 ไบต์ มาแสดงตามลำดับรหัส ASCII ตั้งแต่รหัส 32 ถึง 255 ไฟล์ที่ใช้ได้กับโปรแกรมนี้คือ FONTS.LQ, NORMALS.P24 และ ITALICS.P24

โปรแกรม 10.9

```
PROGRAM DisplayFont8064Byte;
USES Graph;
TYPE CharByte      = ARRAY[1..36] OF Byte;
     RecordOfChar  = RECORD
                       OneChar : CharByte;
                     END;
VAR GraphDriver,GraphMode : Integer;
    FontFile   : File OF RecordOfChar;
    FileName   : STRING[14];
    CharRec    : RecordOfChar;
    Ascii      : ARRAY [32..255] OF CharByte;
    Row,Col    : Integer;
    CharNo     : Byte;

PROCEDURE ReadFont;
VAR CharNo : Byte;
BEGIN
   Assign(FontFile,FileName);
   Reset(FontFile);
   CharNo := 32;
   WHILE NOT Eof(FontFile) DO
   BEGIN
      Read(FontFile,CharRec);
      Ascii[CharNo] := CharRec.OneChar;
      Inc(CharNo);
   END;
   Close(FontFile);
END;
```

```
PROCEDURE SwapBit(VAR NewBit : Byte);
VAR I,X,Y,Z : Integer;
BEGIN
  Z := 1; Y := NewBit; NewBit := 0;
  FOR I := 1 TO 8 DO
  BEGIN
    X := Y DIV (128 DIV Z);
    X := X * Y;
    Y := Y MOD (128 DIV Z);
    Z := Z * 2;
    NewBit := NewBit + X
  END
END;

PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
   StartY : Integer;
BEGIN
  StartY := Y;
  FOR ByteNo :=1 TO 36 DO
  BEGIN
    IF GrDriver <> 7 THEN SwapBit(Ascii[CharNo,ByteNo]);
    SetLineStyle(UserBitLn,Ascii[CharNo,ByteNo],1);
    Line(X,Y, X,Y-15);
    Inc(Y,8);
    IF ByteNo MOD 2 = 0 THEN
    BEGIN
      Y := StartY;
      Inc(X)
    END;
```

```
  END;
END;

BEGIN
   WriteLn('DISPLAY FONT FROM FILE 8064 BYTE');
   Write('Enter FONTS.LQ, NORMALS.P24 or ITALICS.P24 : ');
   ReadLn(FileName);
   ReadFont;
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   ClearDevice;
   Row     := 30;
   Col     := 1;
   CharNo  := 32;
   REPEAT
      DisplayChar(Col,Row,CharNo);
      Inc(CharNo);
      Inc(Col,30);
      IF Col > GetMaxX - 30 THEN
      BEGIN
         Col := 1;
         Inc(Row,30)
      END;
   UNTIL CharNo = 255;
   ReadLn;
   CloseGraph
END.
```



รูปที่ 10.9 อักษรจากไฟล์ NORMALS.P24

ตัวอย่าง โปรแกรม 10.10 ใช้สำหรับแสดงอักษรบนจอภาพโดยนำอักษรสำหรับเครื่องพิมพ์ทุกตัวที่มีอยู่ในไฟล์ขนาด 8064 ไบต์ มาแสดงตามลำดับรหัส ASCII ตั้งแต่รหัส 32 ถึง 255 ไฟล์ที่ใช้ได้กับโปรแกรมนี้ คือ FONTA1.NLQ และ FONTA2.NLQ โปรดสังเกตว่าการกำหนดลักษณะเส้นจะต้องทำโดยการแบ่งความยาวของเส้นจาก 36 ให้เหลือครึ่งหนึ่งคือ 18 แล้วนำข้อมูลจากแต่ละส่วนมา OR กันตามลำดับไบต์ต่อไบต์ ดังสเตตเมนต์ต่อไปนี้

```
SetLineStyle(UserBitLn, Ascii[CharNo,ByteNo]
                 OR Ascii[CharNo,ByteNo+18], 1)
```

เหตุที่ต้องทำเช่นนี้เนื่องจากเป็นฟอนต์ชนิด NLQ(Near Letter Quality) ซึ่งเครื่องพิมพ์จะพิมพ์เส้นละ 2 ครั้งให้เหลื่อมกันเล็กน้อย เพื่อปิดช่องว่างระหว่างจุดให้เห็นเป็นเส้นทึบ ดังนั้น ในการนำข้อมูลชนิดนี้มาแสดงที่จอภาพเรา จึงต้องนำข้อมูลแต่ละคู่มา OR กันเพื่อให้เห็นเป็นเส้นทึบเช่นเดียวกัน

โปรแกรม 10.10

```
PROGRAM DisplayFont8064ByteRW;
USES Graph;
TYPE CharByte       = ARRAY[1..36] OF Byte;
     RecordOfChar  = RECORD
                       OneChar : CharByte;
                     END;
VAR GraphDriver,GraphMode : Integer;
    FontFile   : File OF RecordOfChar;
    FileName   : STRING[14];
    CharRec    : RecordOfChar;
    Ascii      : ARRAY [32..255] OF CharByte;
    Row,Col    : Integer;
    CharNo     : Byte;

PROCEDURE ReadFont;
VAR CharNo : Byte;
BEGIN
  Assign(FontFile,FileName);
  Reset(FontFile);
  CharNo := 32;
  WHILE NOT Eof(FontFile) DO
  BEGIN
     Read(FontFile,CharRec);
     Ascii[CharNo] := CharRec.OneChar;
     Inc(CharNo);
  END;
  Close(FontFile);
END;
```

```
PROCEDURE SwapBit(VAR NewBit : Byte);
VAR I,X,Y,Z : Integer;
BEGIN
   Z := 1; Y := NewBit; NewBit := 0;
   FOR I := 1 TO 8 DO
   BEGIN
      X := Y DIV (128 DIV Z);
      X := X * Y;
      Y := Y MOD (128 DIV Z);
      Z := Z * 2;
      NewBit := NewBit + X
   END
END;

PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
    StartY : Integer;
BEGIN
  StartY := Y;
  FOR ByteNo :=1 TO 18 DO    {36/2}
  BEGIN
     IF GrDriver <> 7 THEN
     BEGIN
        SwapBit(Ascii[CharNo,ByteNo]);
        SwapBit(Ascii[CharNo,ByteNo+8])
     END;
     SetLineStyle(UserBitLn,Ascii[CharNo,ByteNo]
                             OR Ascii[CharNo,ByteNo+18],1);
     Line(X,Y, X,Y-15);
     Inc(Y,8);
```

```
      Y := StartY;
      Inc(X)
   END;
END;
BEGIN
   WriteLn('DISPLAY RW FONT FROM FILE 8064 BYTE');
   Write('Enter FONTA1.NLQ or FONTA2.NLQ : ');
   ReadLn(FileName);
   ReadFont;
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   ClearDevice;
   Row     := 30;
   Col     := 1;
   CharNo  := 32;
   REPEAT
      DisplayChar(Col,Row,CharNo);
      Inc(CharNo);
      Inc(Col,30);
      IF Col > GetMaxX - 30 THEN
      BEGIN
         Col := 1;
         Inc(Row,30)
      END;
   UNTIL CharNo = 255;
   ReadLn;
   CloseGraph
END.
```

ตัวอย่าง โปรแกรม 10.11 ใช้สำหรับแสดงอักษรบนจอภาพโดยนำอักษรสำหรับเครื่องพิมพ์ทุกตัวที่มีอยู่ในไฟล์ขนาด 11264 ไบต์ และ 9856 ไบต์มาแสดงตามลำดับรหัส ASCII ตั้งแต่รหัส 0 ถึง 255 ไฟล์ที่ใช้ได้กับโปรแกรมนี้คือ FONT1.NLQ, FONT2.NLQ, FONTS.NLQ, NORMAL.PRN, NORMALS.PRN และ ITALIC.PRN โปรดสังเกตว่าการกำหนดลักษณะเส้นจะต้องทำโดยการแบ่งความยาวของเส้นจาก 44 ให้เหลือครึ่งหนึ่งคือ 22 แล้วนำข้อมูลจากแต่ละส่วนมา OR กันตามลำดับไบต์ต่อไบต์ ดังสเตตเมนต์ต่อไปนี้

```
SetLineStyle(UserBitLn, Ascii[CharNo,ByteNo]
                        OR Ascii[CharNo,ByteNo+22], 1)
```

เหตุที่ต้องทำเช่นนี้เนื่องจากเป็นฟอนต์ชนิด NLQ ซึ่งเครื่องพิมพ์จะพิมพ์เส้นละ 2 ครั้งให้เหลื่อมกันเล็กน้อย เพื่อปิดช่องว่างระหว่างจุดให้เห็นเป็นเส้นทึบ ดังนั้น ในการนำข้อมูลชนิดนี้มาแสดงที่จอภาพ เราจึงต้องนำข้อมูลแต่ละคู่มา OR กันเพื่อให้เห็นเป็นเส้นทึบเช่นเดียวกัน

โปรแกรม 10.11

```
PROGRAM DisplayFont11264And9856Byte;
USES Graph;
TYPE CharByte       = ARRAY[1..44] OF Byte;
     RecordOfChar   = RECORD
                        OneChar : CharByte;
                      END;
VAR GraphDriver,GraphMode : Integer;
    FontFile   : File OF RecordOfChar;
    FileName   : STRING[14];
    CharRec    : RecordOfChar;
    Ascii      : ARRAY [0..255] OF CharByte;
```

```
  Row,Col     : Integer;
  CharNo      : Byte;

PROCEDURE ReadFont;
VAR CharNo : Byte;
BEGIN
   Assign(FontFile,FileName);
   Reset(FontFile);
   CharNo := 0;
   WHILE NOT Eof(FontFile) DO
   BEGIN
      Read(FontFile,CharRec);
      Ascii[CharNo] := CharRec.OneChar;
      Inc(CharNo);
   END;
   Close(FontFile);
END;
PROCEDURE SwapBit(VAR NewBit : Byte);
VAR I,X,Y,Z : Integer;
BEGIN
   Z := 1; Y := NewBit; NewBit := 0;
   FOR I := 1 TO 8 DO
   BEGIN
      X := Y DIV (128 DIV Z);
      X := X * Y;
      Y := Y MOD (128 DIV Z);
      Z := Z * 2;
      NewBit := NewBit + X
```

```
  END;
END;
PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
    StartY : Integer;
BEGIN
  StartY := Y;
  FOR ByteNo :=1 TO 22 DO
  BEGIN
    IF GrDriver <> 7 THEN
    BEGIN
       SwapBit(Ascii[CharNo,ByteNo]);
       SwapBit(Ascii[CharNo,ByteNo + 22])
    END;
    SetLineStyle(UserBitLn,
                (Ascii[CharNo,ByteNo] OR Ascii
                 [CharNo,ByteNo+22]),1);
    Line(X,Y, X,Y-15);
    Inc(Y,1);
    Y := StartY;
    Inc(X)
  END;
END;

BEGIN
  WriteLn('DISPLAY FONT FROM FILE 9856 AND 11264 BYTE');
  WriteLn('Enter FONT1.NLQ, FONT2.NLQ, FONTS.NLQ, NORMAL.PRN');
  Write('         NORMALS.PRN or ITALIC.PRN : ');
```

```
  ReadLn(FileName);
  ReadFont;
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  ClearDevice;
  Row     := 30;
  Col     := 1;
  CharNo := 32;
  REPEAT
    DisplayChar(Col,Row,CharNo);
    Inc(CharNo);
    Inc(Col,30);
    IF Col > GetMaxX - 30 THEN
    BEGIN
      Col := 1;
      Inc(Row,30)
    END;
  UNTIL CharNo = 255;
  ReadLn;
  CloseGraph
END.
```

10.8 โปรแกรมสร้างฟอนต์สำหรับจอภาพ

ฟอนต์ในไฟล์ต่างๆ ที่เราได้ศึกษาผ่านมา ต่างก็เป็นฟอนต์ซึ่งสร้างขึ้นด้วยโปรแกรมสร้างฟอนต์ ตามปกติโปรแกรมสร้างฟอนต์จะแบ่งเป็น 2 กลุ่มคือ โปรแกรมสร้างฟอนต์สำหรับจอภาพและโปรแกรมสร้างฟอนต์สำหรับเครื่องพิมพ์ ในหัวข้อนี้จะได้กล่าวถึงการเขียนโปรแกรมสร้างฟอนต์สำหรับจอภาพ ซึ่งตั้งชื่อโปรแกรมเป็นฟอนต์814 (FONT814)

10.8.1 คุณสมบัติของโปรแกรมฟอนต์ 814 โปรแกรมฟอนต์814 เป็นโปรแกรมสร้างฟอนต์ ซึ่งมีคุณสมบัติดังนี้

1. สร้างฟอนต์ชนิดบิตแมน
2. ตารางอักษรมีขนาด 8 x 14
3. ในฟอนต์ประกอบด้วยอักษร 224 ตัว เรียงตามรหัส ASCII จาก 32 ถึง 255 ดังแสดงในตารางที่ 10.4

ตารางที่ 10.4 อักษรและรหัส ASCII ที่สร้างด้วยโปรแกรมฟอนต์ 814

ASCII	อักษร	ASCII	อักษร	ASCII	อักษร	ASCII	อักษร	ASCII	อักษร
32		77	M	122	z	167	ฉ	212	ไ
33	!	78	N	123	{	168	ช	213	ๆ
34	"	79	O	124	¦	169	ซ	214	ฯ
35	#	80	P	125	}	170	ฌ	215	ุ
36	$	81	Q	126	~	171	ญ	216	ู
37	%	82	R	127		172	ฎ	217	ิ
38	&	83	S	128	๐	173	ฏ	218	ี
39	'	84	T	129	๑	174	ฐ	219	ึ
40	(	85	U	130	๒	175	ฑ	220	ื
41	)	86	V	131	๓	176	ฒ	221	ั
42	*	87	W	132	๔	177	ณ	222	
43	+	88	X	133	๕	178	ด	223	์
44	,	89	Y	134	๖	179	ต	224	่
45	-	90	Z	135	๗	180	ถ	225	้
46	.	91	๘	136		181	ท	226	๊
47	/	92	๙	137		182	ธ	227	๋

ตารางที่ 10.4 (ต่อ) อักษรและรหัส ASCII ที่สร้างด้วยโปรแกรมฟอนต์ 814

ASCII	อักษร	ASCII	อักษร	ASCII	อักษร	ASCII	อักษร	ASCII	อักษร
48	0	93	]	138		183	น	228	์
49	1	94	^	139		184	บ	229	
50	2	95	_	140		185	ป	230	
51	3	96	'	141		186	ผ	231	
52	4	97	a	142	ๆ	187	ฝ	232	
53	5	98	b	143	┼	188	พ	233	
54	6	99	c	144	┴	189	ฟ	234	
55	7	100	d	145	┬	190	ภ	235	
56	8	101	e	146	┤	191	ม	236	
57	9	102	f	147	├	192	ย	237	
58	:	103	g	148		193	ร	238	
59	;	104	h	149	┌	194	ฤ	239	
60	<	105	i	150	┬	195	ล	240	
61	=	106	j	151	┐	196	ว	241	
62	>	107	k	152	├	197	ศ	242	
63	?	108	l	153	┼	198	ษ	243	
64	@	109	m	154	┤	199	ส	244	
65	A	110	n	155	└	200	ห	245	
66	B	111	o	156	┴	201	ฬ	246	
67	C	112	p	157	┘	202	อ	247	
68	D	113	q	158	─	203	ฮ	248	
69	E	114	r	159	│	204	ะ	249	
70	F	115	s	160		205		250	
71	G	116	t	161	ก	206	า	251	

ตารางที่ 10.4 (ต่อ) อักษรและรหัส ASCII ที่สร้างด้วยโปรแกรมฟอนต์ 814

ASCII	อักษร	ASCII	อักษร	ASCII	อักษร	ASCII	อักษร	ASCII	อักษร
72	H	117	u	162	ข	207	252	ำ	
73	I	118	v	163	ค	208	253	เ	
74	J	119	w	164	ฆ	209	254	แ	
75	K	120	x	165	ง	210	255	โ	
76	L	121	y	166	จ	211	256	ใ	

10.8.2 ส่วนประกอบของโปรแกรมฟอนต์ 814 เพื่ออำนวยความสะดวกแก่ผู้ใช้โปรแกรมฟอนต์814 จึงต้องออกแบบโปรแกรมให้สามารถดำเนินการเกี่ยวกับอักษรในด้านต่างๆต่อไปนี้ได้คือ สร้างอักษร แก้ไขเปลี่ยนแปลงอักษร และแสดงฟอนต์ทั้งหมดพร้อมกันในโหมดกราฟิกส์ ซึ่งแต่ละส่วนมีรายละเอียดดังต่อไปนี้

1. ส่วนสร้างแก้ไขและบันทึกตัวอักษร ทำหน้าที่สร้างตัวอักษรชนิดบิตแมพ โดยการสร้างจุดให้อยู่ในตำแหน่งต่างๆ ของตารางขนาด 8 x 14 เพื่อให้มองเห็นเป็นอักษรตามต้องการ สำหรับอักษรที่มีอยู่แล้วในไฟล์ โปรแกรมสามารถเรียกมาแก้ไขเปลี่ยนแปลงให้เป็นอักษรใหม่หรือแบบใหม่ได้ เมื่อได้อักษรในรูปแบบที่ต้องการแล้วสามารถสั่งให้โปรแกรมนำอักษรทั้งหมดหรือเพียงบางตัวบันทึกไว้ในไฟล์ได้

2. ส่วนแสดงฟอนต์ ทำหน้าที่นำอักษรทุกตัวที่มีอยู่ในไฟล์มาแสดงที่จอภาพ โดยแสดงเรียงลำดับตามรหัส ASCII จากรหัส 32 ถึง 255

10.8.3 การทำงานของโปรแกรมฟอนต์ 814 โปรแกรมจะเริ่มโดยแสดงรายการให้เลือกดังแสดงในรูปที่ 10.10

```
===== FONT814N =====
Version 1.0

------------------------------------------------

Program for Create, Edit and Display Fonts.
Limitation :
                              ASCII 32 to 255.
                             Font size 8 x 14.
================================================

E : Edit Font
D : Display Fonts in Graphics mode
Q : Qxit

================================================

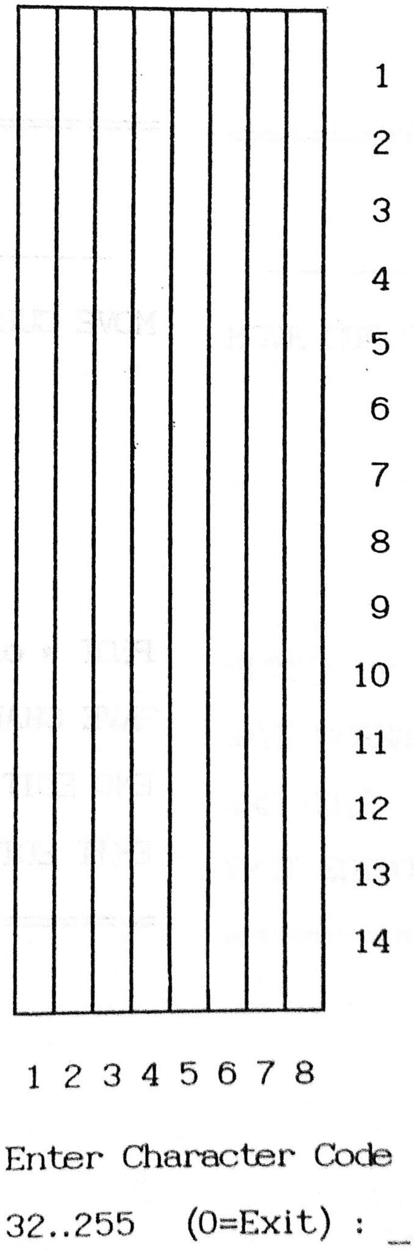
For select : Press E D or Q
```

รูปที่ 10.10 จอภาพแรกของโปรแกรมฟอนต์ 814 ซึ่งมีรายการให้เลือก 3 รายการ

แต่รายการมีความหมายดังนี้

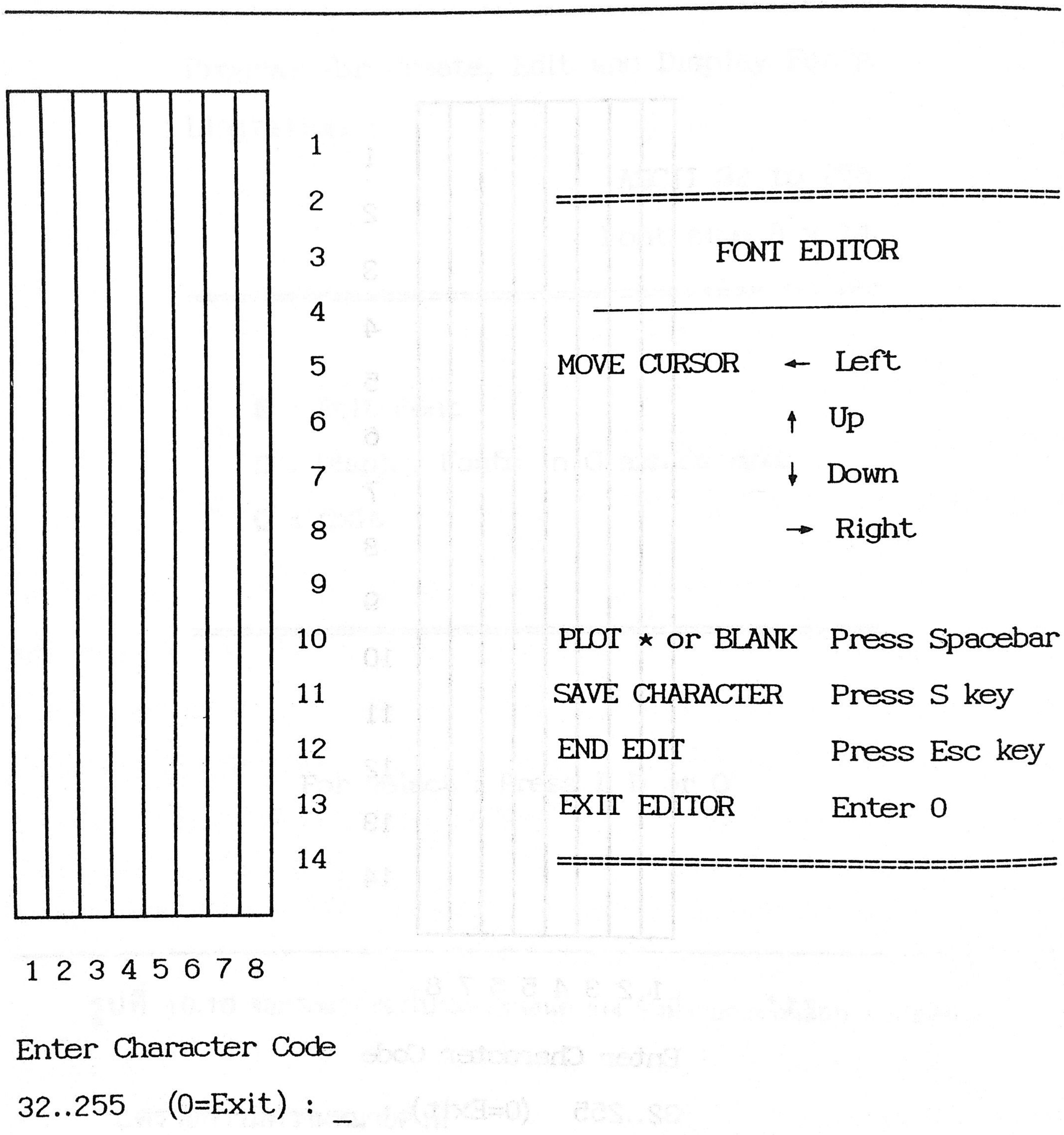
1. รายการ E (edit font) ใช้สำหรับดำเนินการเกี่ยวกับ การสร้าง แก้ไขและบันทึกตัวอักษรเข้าในไฟล์ที่ได้สร้างไว้แล้ว เมื่อเลือกรายการนี้ โปรแกรมจะ

เริ่มด้วยการถามชื่อไฟล์ ถ้าชื่อไฟล์ที่ป้อนเข้าไปไม่มีอยู่ในดิสก์ โปรแกรมจะสร้างไฟล์นั้นขึ้นและดำเนินการต่อไป สำหรับไฟล์ที่มีอยู่แล้วโปรแกรมจะอ่านข้อมูลอักษรทั้งหมดเข้ามาในหน่วยความจำและรอรับรหัสอักษรที่จะสร้างหรือแก้ไข โดยแสดงจอภาพดังรูปที่ 10.11



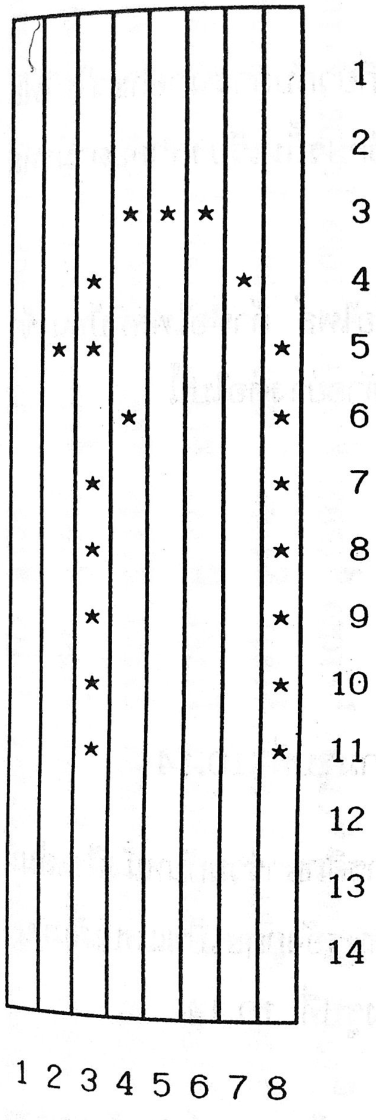
รูปที่ 10.11 จอภาพสำหรับป้อนรหัสอักษรที่ต้องการสร้างหรือแก้ไข

เมื่อป้อนรหัสอักษรเป็นตัวเลขค่าใดค่าหนึ่ง ตั้งแต่ 32 ถึง 255 ถ้ารหัสนั้นมีอักษรอยู่แล้ว โปรแกรมจะแสดงอักษรนั้นในตาราง สำหรับรหัสที่ไม่มีอักษรซึ่งอาจจะเนื่องจากยังไม่ได้สร้างขึ้น หรือเนื่องจากเป็นไฟล์ใหม่ โปรแกรมจะแสดงคำอธิบายวิธีการใช้คีย์ต่างๆ ซึ่งที่จอภาพจะมีลักษณะดังรูปที่ 10.12



รูปที่ 10.12 จอภาพสำหรับการสร้าง แก้ไขและบันทึกตัวอักษร

หน้าที่หลักของโปรแกรมในตอนนี้ก็คือ การสร้างเครื่องหมาย ⋆ ให้เกิดขึ้นที่ตำแหน่งต่างๆ ในตารางเพื่อให้มองเห็นเป็นอักษรหนึ่งตัว ซึ่งมีวิธีการสร้างง่ายๆ โดยเลื่อนเคอร์เซอร์ด้วยคีย์ลูกศรไปยังตำแหน่งที่ต้องการ แล้วกดคีย์วรรค (spacebar) เพื่อสร้างเครื่องหมาย ⋆ คีย์วรรคจะทำหน้าที่ได้ทั้งการสร้างและลบเครื่องหมาย ⋆




```
==============================
        FONT EDITOR
------------------------------
MOVE CURSOR   ←  Left
              ↑  Up
              ↓  Down
              →  Right

PLOT * or BLANK   Press Spacebar
SAVE CHARACTER    Press S key
END EDIT          Press Esc key
EXIT EDITOR       Enter 0
==============================

Enter Character Code
32..255   (0=Exit) : 161
```

รูปที่ 10.13 อักษร ก แบบตัวตรงเส้นบาง

โดยมีหลักว่าถ้ากดคีย์วรรคตอนเคอร์เซอร์อยู่ที่ตำแหน่งว่างจะเกิดเครื่องหมาย ★ ขึ้น และถ้ากดคีย์วรรคตอนเคอร์เซอร์อยู่ที่เครื่องหมาย ★ จะเป็นการลบ ★ ให้เป็นว่าง แสดงตัวอย่างรูปร่างอักษร ก ซึ่งรหัส ASCII เป็น 161 ดังรูปที่ 10.13

เมื่อได้รูปแบบอักษรที่ต้องการแล้ว ถ้าต้องการบันทึกไว้ในไฟล์ให้กดคีย์ S เมื่อต้องการเลิกสร้างและแก้ไขอักษรที่กำลังดำเนินการอยู่ให้กดคีย์ Esc สำหรับการหยุดสร้างและแก้ไขอักษรเพื่อกลับไปจอภาพเริ่มต้น ให้ป้อนรหัสอักษรเป็นเลข 0

2. รายการ D (display font) ดำเนินการเกี่ยวกับการนำอักษรในไฟล์มาแสดงที่จอภาพ โดยแสดงทั้งอักษรและรหัส ASCII ของอักษรนั้นเรียงลำดับตามรหัส ASCII จาก 32 ถึง 255

เมื่อเลือกรายการนี้ โปรแกรมจะเริ่มด้วยการถามชื่อไฟล์ ถ้าชื่อไฟล์ที่ป้อนเข้าไปไม่มีอยู่ในดิสก์ โปรแกรมจะแสดงข้อความไม่พบไฟล์ ดังตัวอย่างต่อไปนี้

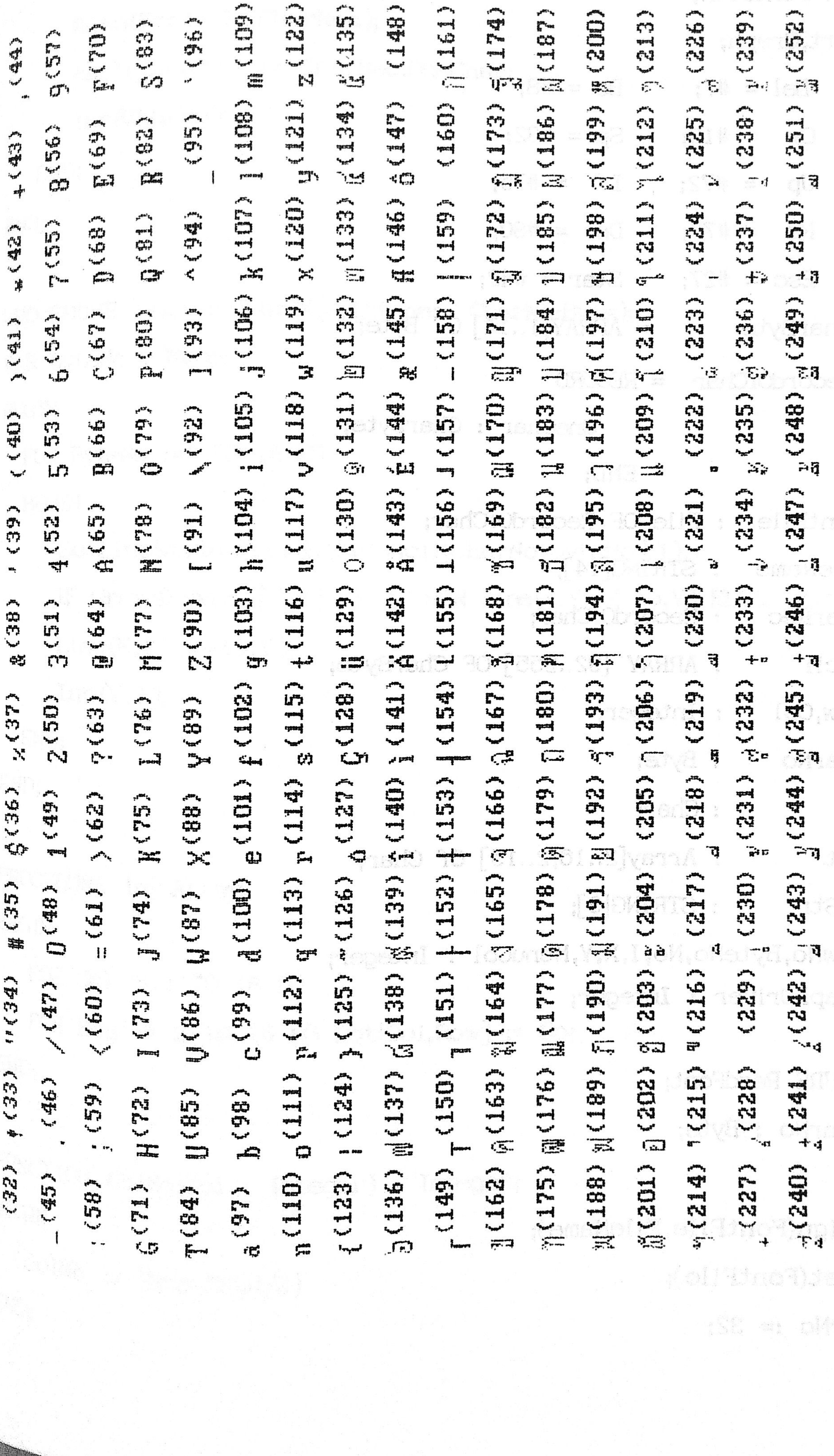
```
File XXX Not found
Press ENTER
```

โดย XXX คือ ชื่อไฟล์ที่เราป้อนเข้าไป

ถ้ามีไฟล์อักษรอยู่ในดิสก์โปรแกรมจะแสดงจอภาพตามรูปที่ 10.14

ในดิสก์ 157 โปรแกรมซึ่งสร้างขึ้นสำหรับช่วยในการศึกษาร่วมกับหนังสือเล่มนี้ มีไฟล์อักษรชื่อ 814.FON ซึ่งเป็นฟอนต์อักษรแบบตัวตรง อักษรอังกฤษเป็นแบบเส้นหนา อักษรไทยเป็นแบบเส้นบาง เมื่อแสดงที่จอภาพจะมีลักษณะดังรูปที่ 10.14

10.8.4 โปรแกรมสร้างฟอนต์ 814 โปรแกรมสร้างฟอนต์สำหรับจอภาพขนาดอักษร 8 x 14 ตั้งชื่อโปรแกรมเป็น FONT814 มีรายละเอียดดังแสดงในโปรแกรม 10.12



รูปที่ 10.14 ฟอนต์ขนาด 8×14 จากไฟล์ 814.FON

โปรแกรม 10.12

```
PROGRAM Font814N;
USES Crt,Graph;
CONST    Bel = #7;       Bs = #8;
         Cr  = #13;      Sp = #32;
         Up  = #72;      Lf  = #75;
         Rg  = #77;      Dn  = #80;
         Esc = #27;      Star = #42;
TYPE CharByte        = ARRAY[1..14] OF Byte;
     RecordOfChar  = RECORD
                         OneChar : CharByte;
                     END;
VAR FontFile   : File OF RecordOfChar;
    FileName   : STRING[14];
    CharRec    : RecordOfChar;
    Ascii      : ARRAY [32..255] OF CharByte;
    Row,Col    : Integer;
    CharNo     : Byte;
    Ch         : Char;
    Dot        : Array[2..16,2..15] Of Char;
    NoStr      : STRING[3];
    RowNo,ByteNo,No,I,X,Y,MenuCol : Integer;
    GraphDriver : Integer;

PROCEDURE ReadFont;
VAR CharNo : Byte;
BEGIN
   Assign(FontFile,FileName);
   Reset(FontFile);
   CharNo := 32;
```

```
  WHILE NOT Eof(FontFile) DO
  BEGIN
    Read(FontFile,CharRec);
    Ascii[CharNo] := CharRec.OneChar;
    Inc(CharNo);
  END;
END;

PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
BEGIN
  FOR ByteNo :=1 TO 14 DO
  BEGIN
    SetLineStyle(UserBitLn,Ascii[CharNo,ByteNo],1);
    IF GraphDriver {VGA} <> 7 THEN Line(X,Y, X-15,Y) ELSE {Herc}
    Line(X,Y, X+15,Y);
    Inc(Y,1);
  END;
END;

PROCEDURE InitArray;
BEGIN
  FOR Col := 1 TO 16 DO
  FOR Row := 2 TO 15 DO Dot[Col,Row] := Sp;
END;

FUNCTION ColNo(Col : Integer) : Integer;
BEGIN
  ColNo := Round(Col/2)
END;
```

```
PROCEDURE Condition(Low:Integer;Var X:Integer;High:Integer);
BEGIN
   If X < Low Then X := Low Else
   If X > High Then X := High
END;

PROCEDURE SetXY(NewX,NewY : Integer);
BEGIN
   Condition(2,NewX,16); {left & right border}
   Condition(2,NewY,15); {top & bottom border}
   GotoXY(NewX,NewY);
END;

PROCEDURE WriteFile;
BEGIN
   Reset(FontFile);
   Seek(FontFile,No-32); Ascii[No] := CharRec.OneChar;
   Write(FontFile,CharRec)
END;

FUNCTION TwoPowerN(N : Byte) : Byte;
VAR X,Power : Byte;
BEGIN
   IF N = 0 THEN TwoPowerN := 1 ELSE
   BEGIN
      X := 1;
      FOR Power := 1 TO N DO X := 2 * X;
      TwoPowerN := X
   END
END;
```

```
PROCEDURE HandleKey(Ch : Char);
VAR M,Power : Byte;
BEGIN
  IF Ch = Sp THEN
  BEGIN
    IF Dot[WhereX,WhereY] = Sp THEN
    BEGIN
      Write(Star);
      SetXY(WhereX-1,WhereY);
      Dot[WhereX,WhereY] := Star
    END ELSE
    BEGIN
      Write(Sp);
      SetXY(WhereX-1,WhereY);
      Dot[WhereX,WhereY] := Sp
    END;
  END  ELSE
  IF (Ch ='S') OR (Ch = 's') THEN
  BEGIN
    FOR Row := 2 TO 15 DO
    BEGIN
      Col := 2;  M := 0; Power := 7;
      WHILE Col <= 16 DO
      BEGIN
        IF Dot[Col,Row] = Star THEN M := M + TwoPowerN(Power);
        Inc(Col,2); Dec(Power);
      END;
      CharRec.OneChar[Row-1] := M;
    END;
```

```
    WriteFile;
  END;
END;

PROCEDURE HandleFuncKey(Ch:Char);
BEGIN
  CASE Ch OF
    Up  :  SetXY(WhereX   ,WhereY-1);
    Lf  :  SetXY(WhereX-2,WhereY);
    Rg  :  SetXY(WhereX+2,WhereY);
    Dn  :  SetXY(WhereX   ,WhereY+1)
  END;
END;

PROCEDURE Frame;
BEGIN
  WriteLn(#218,#196,#194,#196,#194,#196,#194,#196,#194,
          #196,#194,#196,#194,#196,#194,#196,#191);
  FOR Row := 1 TO 14 DO
  BEGIN
    FOR Col := 1 TO 9 DO Write(#179,' ');
    WriteLn(Row:2);
  END;
  Write(#192,#196,#193,#196,#193,#196,#193,#196,#193,
        #196,#193,#196,#193,#196,#193,#196,#217); WriteLn;
  WriteLn(' 1 2 3 4 5 6 7 8');
END;

FUNCTION Digit(N : Byte) : Char;
```

```
BEGIN
   IF N < 1   THEN Digit := Chr(Ord(' ') + N)
           ELSE Digit := Chr(Ord('A') + ((N - 66) + 42{char_code}))
END;

PROCEDURE ConvertBin(Number : Integer; VAR X,Y : Word);
VAR I : Integer;
BEGIN
   FOR I := 0 TO 15  DO
   BEGIN
      IF I > 7 THEN
      BEGIN
         GoToXY(X,Y);
         Dot[Round((I-7)*2),RowNo+1] := Digit(Ord(Number < 0));
         Write(Dot[X,Y]);
         Inc(X,2);
      END;
      Number := Number ShL 1
   END;
END;

PROCEDURE OpenGraph;
VAR
   GrDriver,GrMode :integer;
BEGIN
   GrDriver := Detect;
   InitGraph(GrDriver, GrMode, '');
   ClearDevice
END;
```

```
PROCEDURE Direction;
BEGIN
  window(40,3,80,20);
  WriteLn('================================');
  WriteLn('              FONT EDITOR');
  WriteLn('--------------------------------');
  writeLn('MOVE CURSOR ','  ',#27,#45,' Left');
  WriteLn('            ','  ',#24,' Up');
  WriteLn('            ','  ',#25,' Down');
  WriteLn('            ','  ',#45,#26,' Right');WriteLn;
  WriteLn('PLOT * or BLANK  Press Spacebar');
  WriteLn('SAVE CHARACTER   Press S key');
  WriteLn('END EDIT         Press Esc key');
  WriteLn('EXIT EDITOR      Enter 0');
  WriteLn('================================');
  Window(1,1,80,25);
END;

PROCEDURE DisplayDot(X,Y : Word);
BEGIN
  FOR RowNo := 1 TO 14 DO
  BEGIN
    ConvertBin(Ascii[no,RowNo],X,Y);
    IF X > 16 THen BEGIN X := 2; Inc(Y); END;
  END;
  Direction;
  GotoXY(2,2);
  Repeat
    Ch := ReadKey;
```

```
    IF Ch <> #0 THEN HandleKey(Ch)
                ELSE HandleFuncKey(ReadKey);
  UNTIL Ch IN [Esc,'S','s'];
END;

PROCEDURE CreateFile;
BEGIN
  ReWrite(FontFile);
  FOR No := 32 TO 255 DO
  BEGIN
    FOR ByteNo := 1 TO 14 DO CharRec.OneChar[ByteNo] := 0;
    Write(FontFile,CharRec)
  END;
  Close(FontFile);
END;

BEGIN
  DirectVideo := True;
  ClrScr;
  MenuCol := 58;
  REPEAT
    ClrScr;
    WriteLn('        =====  FONT814N  =====               ':MenuCol);
    WriteLn('               Version 1.0                   ':MenuCol);
    WriteLn;
    WriteLn('---------------------------------------------':MenuCol);
    WriteLn('Program for Create, Edit and Display Fonts.':MenuCol);
    WriteLn('Limitation :                                ':MenuCol);
    WriteLn('                         ASCII 32 to 255.':MenuCol);
    WriteLn('                        Font size 8 x 14.':MenuCol);
```

```
WriteLn('==========================================':MenuCol);
WriteLn;
WriteLn('      E : Edit Font                            ':MenuCol);
WriteLn('      D : Display Fonts in Graphics mode       ':MenuCol);
WriteLn('      Q : Qxit                                 ':MenuCol);
WriteLn;
WriteLn('==========================================':MenuCol);
WriteLn;
WriteLn('         For select : Press E D or Q          ':MenuCol);
GotoXY(38,18);  Ch := ReadKey;
CASE Ch OF
    'E','e' : BEGIN
                   GotoXY(1,17); ClrEol; GotoXY(28,17);
                   WriteLn('Edit Font');
                   GotoXY(1,18);clrEol;  GotoXY(28,18);
                   Write('Enter File Name : '); ReadLn(FileName);
                   IF Length(FileName) > 0 THEN
                   BEGIN
                     ClrScr;Frame;
                     Assign(FontFile,FileName);
                     {$I-} Reset(FontFile); {$I+}
                     IF IoResult <> 0 THEN CreateFile;
                     ReadFont;
                     REPEAT
                       repeat
                           gotoxy(1,19);
                           WriteLn('Enter Character Code');
                           Write('32..255   (0=Exit) : ');
                           ClrEol;ReadLn(NoStr);
                           Val(NoStr,No,I);
```

```
            UNTIL I = 0;
            IF No IN[32..255] THEN DisplayDot(2,2);
         UNTIL No = 0;
         Close(FontFile);
      END;
   END;
'D','d' : BEGIN
      GotoXY(1,17); ClrEol; GotoXY(28,17);
      WriteLn('Display Font in Graphics mode');
      GotoXY(1,18);clrEol; GotoXY(28,18);
      Write('Enter File Name : '); ReadLn(FileName);
      IF Length(FileName) > 0 THEN
      BEGIN
         Assign(FontFile,FileName);
         {$I-} Reset(FontFile); {$I+}
         IF IoResult <> 0 THEN
         BEGIN
            ClrScr;
            WriteLn('File ',FileName,' Not found');
            Write('Press ENTER '); ReadLn;
         END ELSE
         BEGIN
            ReadFont;Close(FontFile);
            OpenGraph;
            X := 8; Y := 0;
            FOR I := 32 TO 255 DO
            BEGIN
               IF X > 660 THEN BEGIN X := 8; Y:=Y+20 EN;
               DisplayChar(X,Y,I); Inc(X,10);
               Str(I,NoStr);OutTextXY(X,Y,'('+NoStr+')');
```

```
                    Inc(X,56);
                   END;
                   REPEAT UNTIL KeyPressed;
                 END;
                 CloseGraph
               END;
             END;
     'Q','q' : ;
   END;
  UNTIL Ch IN ['Q','q'];
END.
```

10.9 การแสดงอักษรกราฟิกส์ไทย-อังกฤษจากคีย์บอร์ด

การแสดงอักษรกราฟิกส์ในหัวข้อที่ผ่านมาเป็นการแสดงโดยนำอักษรจากไฟล์มาแสดงที่จอภาพเรียงตามลำดับรหัส ASCII ในหัวข้อนี้จะได้กล่าวถึงวิธีการแสดงอักษรกราฟิกส์ ซึ่งจะแสดงครั้งละหนึ่งอักษรตามคีย์อักษรที่ถูกกด โดยสามารถแสดงได้ทั้งอักษรไทย อังกฤษ และเครื่องหมายต่างๆ ที่ปรากฏอยู่บนคีย์บอร์ด รายละเอียดมีดังต่อไปนี้

10.9.1 อักษรบนคีย์บอร์ด คีย์บอร์ดของคอมพิวเตอร์แบ่งเป็น 2 กลุ่มคือ กลุ่มตัวอักษรและกลุ่มควบคุมการทำงาน ความแตกต่างที่เห็นได้ง่ายของคีย์ทั้งสองกลุ่มนี้ก็คือ ถ้ากดคีย์ในกลุ่มตัวอักษร เราจะเห็นอักษรจากคีย์นั้นปรากฏขึ้นที่จอภาพ 1 ตัวต่อการกด 1 ครั้ง แต่การกดคีย์ในกลุ่มควบคุมการทำงาน เราอาจเห็นอักษร ข้อความ หรือไม่มีอะไรปรากฏที่จอภาพเลยก็ได้ ทั้งนี้ขึ้นอยู่กับโปรแกรมที่ทำงานอยู่ในขณะนั้น

คีย์ในกลุ่มตัวอักษรประกอบด้วยอักษรภาษาอังกฤษ ตัวเลขและเครื่องหมายต่างๆ รวม 95 ตัว โดยใช้รหัส ASC ตั้งแต่ 32 ถึง 126 ดังแสดงในตารางที่ 10.5

ตารางที่ 10.5 อักษรบนคีย์บอร์ดซึ่งแยกได้ 4 ประเภท

อักษร	รหัส ASCII	จำนวนตัว
A – Z	65 - 90	26
a – z	97 - 122	26
0 – 9	48 - 57	10
เครื่องหมายต่างๆ	รหัสที่เหลือจาก 3 รายการข้างบน	33

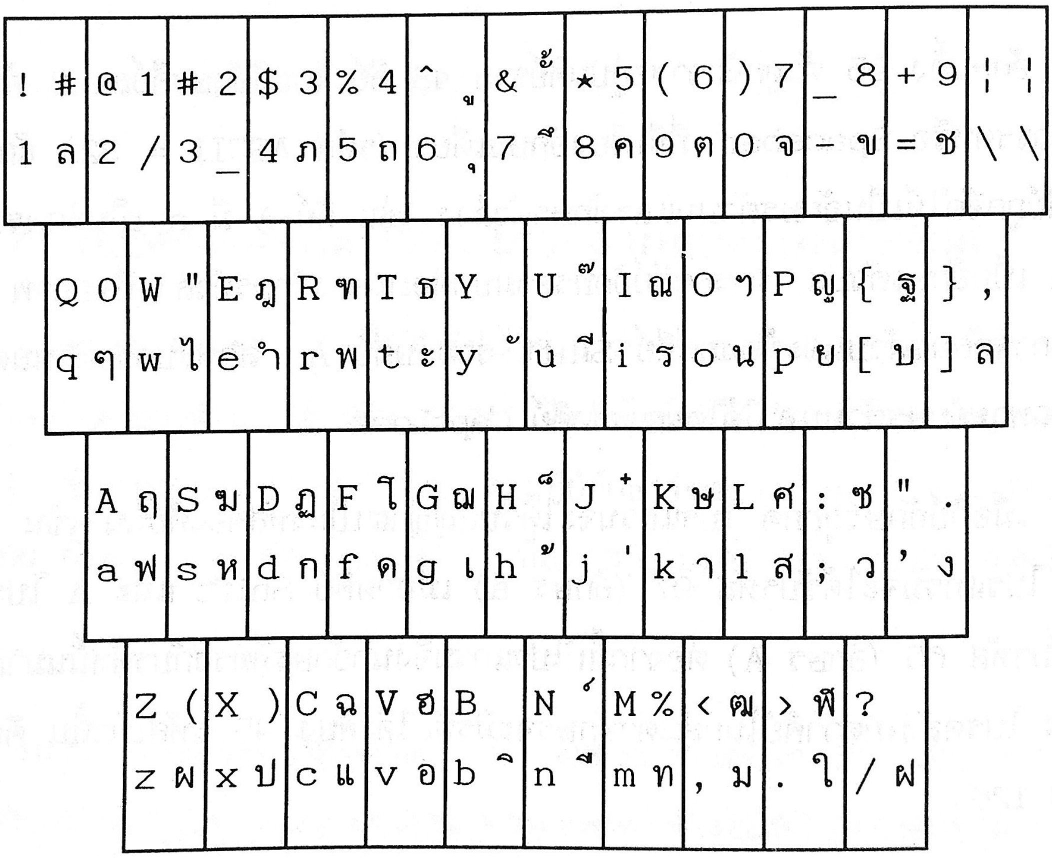
อักษรทั้ง 95 ตัวถูกจัดวางอยู่บนคีย์รวม 48 คีย์ โดยมีอักษรคีย์ละ 2 ตัว ยกเว้นคีย์วรรคหรือ Spacebar ที่มีเพียงอักษรเดียว (รหัส ASCII = 32) อักษร 2 ตัวบนคีย์ถูกจัดให้เป็นอักษรตัวบนและอักษรตัวล่าง เช่น คีย์ A มี a เป็นอักษรตัวล่างและ A เป็นอักษรตัวบน การกดคีย์อักษรตามปกติจะแสดงอักษรตัวล่างที่จอภาพ คือ a ถ้าต้องการอักษรตัวบนจะต้องกดคีย์ Shift ร่วมกับคีย์ A สำหรับการกำหนดคีย์ให้แสดงเฉพาะอักษรตัวบนทำได้โดยการกดคีย์ CapsLock

เมื่อคีย์อักษรถูกกด โปรแกรมจะได้รับสัญญาณเป็นรหัสของคีย์นั้น เช่น เมื่อกดคีย์ A โปรแกรมจะได้รับรหัส 97 (อักษร a) เมื่อกดคีย์ Shift และ A โปรแกรมจะได้รับรหัส 65 (อักษร A) ต่อจากนั้นโปรแกรมจึงนำอักษรที่ตรงกับรหัสนั้นมาแสดงที่จอภาพ โปรดสังเกตว่าคีย์ในกลุ่มตัวอักษรจะมีรหัสได้เพียง 95 รหัสเท่านั้น คือตั้งแต่ 32 ถึง 126

10.9.2 รหัสอักษรไทย-อังกฤษจากคีย์บอร์ด ตามที่เราได้ทราบมาแล้วว่ารหัส ASCII มีจำนวนทั้งหมด 256 รหัส ตั้งแต่รหัส 0 ถึง 255 และมีข้อตกลงว่าห้ามเปลี่ยนแปลงความหมายของรหัสตั้งแต่ 0 ถึง 127 ส่วนที่เหลือให้เปลี่ยนแปลงได้ตามต้องการ ดังนั้นความหมายของ ASCII ตั้งแต่รหัส 128 ถึง 225 จึงอาจจะดัดแปลงให้แตกต่างกันได้ กรณีประเทศไทย เราดัดแปลงรหัสส่วนนี้ให้เป็นอักษรไทย ซึ่ง

ผู้มีความรู้ทางด้านคอมพิวเตอร์ได้พยายามสร้างรหัสอักษรไทยขึ้นหลายแบบ แต่ที่นิยมกันมากมี 2 แบบคือ อักษรไทย แบบสมอ. และอักษรไทยแบบมหาวิทยาลัยเกษตรศาสตร์ (นิยมเรียกย่อว่า รหัสเกษตร) ซึ่งมีแสดงในตารางที่ 10.6

การเรียงอักษรไทยบนคีย์บอร์ดก็มีหลายแบบ แต่ที่รู้จักกันแพร่หลายมีเพียง 2 แบบ คือ แบบเกษมณี และแบบปัตตโชติ ในที่นี้จะใช้คีย์บอร์ดแบบเกษมณีมาเป็นตัวอย่าง อย่างไรก็ตามคีย์บอร์ดแบบเกษมณีของเครื่องพิมพ์ดีดแต่ละเครื่องก็ยังมีความแตกต่างกันบ้างเล็กน้อย ดังนั้นเพื่อให้สะดวกแก่การศึกษา ในที่นี้จึงต้องเลือกคีย์บอร์ดมาหนึ่งแบบ ดังแสดงในรูปที่ 10.15



รูปที่ 10.15 คีย์บอร์ดอักษรไทย-อังกฤษ คีย์อักษรไทยจัดเรียงแบบเกษมณี

จากรูปที่ 10.15 จะเห็นได้ว่าจำนวนคีย์ยังคงมีเท่าเดิมคือ 48 คีย์และได้เพิ่มอักษรไทยเข้าไปอีก 90 ตัว ทำให้คีย์ส่วนใหญ่ คือ 45 คีย์ มีอักษร 4 ตัวต่อหนึ่งคีย์ เช่น คีย์ A มีอักษร a, A, ฟ และ ฤ คีย์ D มีอักษร d, D, ก และ ฎ ซึ่งหมาย

ความว่าคีย์เหล่านี้แต่ละคีย์จะมีรหัส ASCII รวม 4 รหัส เช่น คีย์ A ประกอบด้วย รหัส 97 (อักษร a) 65 (อักษร A) 189 (อักษร ฟ) และ 194 (อักษร ฤ) ดังนั้น ตารางรหัส ASCII ของอักษรทั้งไทยและอังกฤษที่ปรากฏอยู่บนคีย์เดียวกัน และมีลำดับการกดคีย์เหมือนกันจะแสดงได้ดังตารางที่ 10.6

ตารางที่ 10.6 รหัส ASCII ของอักษรไทยและอังกฤษที่อยู่บนคีย์เดียวกัน อักษรไทยเป็นรหัสเกษตร

อังกฤษ	รหัส	ไทย	รหัส	อังกฤษ	รหัส	ไทย	รหัส	อังกฤษ	รหัส	ไทย	รหัส
	32		32	@	64	1	131	`	96	'	96
!	33	#	35	A	65	ฤ	194	a	97	ฟ	189
"	34	.	46	B	66	%	37	b	98	ิ	217
#	35	2	132	C	67	ฉ	167	c	99	แ	209
$	36	3	133	D	68	ฎ	173	d	100	ก	161
%	37	4	134	E	69	ฎ	172	e	101	ำ	207
&	38	ั้	235	F	70	โ	210	f	102	ด	178
'	39	ง	165	G	71	ฌ	170	g	103	เ	208
(	40	6	136	H	72	๊	223	h	104	้	225
)	41	7	137	I	73	ณ	177	i	105	ร	193
*	42	5	135	J	74	๋	227	j	106	่	224
+	43	9	139	K	75	ษ	198	k	107	า	206
,	44	ม	191	L	76	ศ	197	l	108	ส	199
-	45	บ	162	M	77	?	63	m	109	ท	181
.	46	ใ	211	N	78	์	228	n	110	ี	220
/	47	ฝ	187	O	79	ฯ	214	o	111	น	183
0	48	จ	166	P	80	ญ	171	p	112	ย	192
1	49	ล	195	Q	81	0	130	q	113	ๆ	213

ตารางที่ 10.6 (ต่อ) รหัส ASCII ของอักษรไทยและอังกฤษที่อยู่บนคีย์เดียวกัน อักษรไทยเป็นรหัสเกษตร

อังกฤษ	รหัส	ไทย	รหัส	อังกฤษ	รหัส	ไทย	รหัส	อังกฤษ	รหัส	ไทย	รหัส
2	50	-	45	R	82	ฑ	175	r	114	พ	188
3	51	/	47	S	83	ฆ	164	s	115	ห	200
4	52	ภ	190	T	84	ฐ	182	t	116	ะ	204
5	53	ถ	180	U	85	๊	226	u	117	ี	218
6	54	ุ	215	V	86	ฮ	203	v	118	อ	202
7	55	ึ	219	W	87	"	34	w	119	ไ	212
8	56	ค	163	X	88	)	41	x	120	ป	185
9	57	ต	179	Y	89		222	y	121	ั	221
:	58	ซ	169	Z	90	(	40	z	122	ผ	186
;	59	ว	196	[	91	บ	184	{	123	ฐ	174
<	60	ฒ	176	\	92	\	92	¦	124	¦	124
=	61	ช	168	]	93	ล	195	}	125	,	44
>	62	ฬ	201	^	94	ู	216	~	126	~	126
?	63	&	38	_	95	8	138		127		127

เนื่องจากคีย์อักษรมีรหัสตั้งแต่ 32 ถึง 126 เท่านั้น จึงไม่สามารถส่งรหัสอักษรไทยตัวใดๆ ให้แก่โปรแกรมได้ นั่นคือเมื่อกดคีย์ D โปรแกรมจะได้รับรหัส 100 (อักษร d) เสมอไปไม่ใช่รหัส 161 (อักษร ก) ดังนั้นการเปลี่ยนคีย์บอร์ดอักษรอังกฤษให้เป็นอักษรไทย จึงต้องมีโปรแกรมดำเนินการ

10.9.3 โปรแกรมเปลี่ยนคีย์ไทย-อังกฤษ โปรแกรมที่ทำหน้าที่เปลี่ยนคีย์อักษรจากอังกฤษให้เป็นไทยและจากไทยให้เป็นอังกฤษ มีหลักการง่ายๆ คือ จัดเก็บรหัสของอักษรไทยทั้งหมดไว้ในอาร์เรย์ของข้อมูลชนิดไบต์ โดยให้รหัส ASCII ของอักษรบนคีย์นั้นเป็นอินเด็กซ์ของอาร์เรย์ สมมติว่าเราต้องการจัดเก็บรหัสอักษรต่อไปนี้

เก็บรหัสอักษร พ (รหัส 188) ไว้ที่คีย์ r (รหัส 114)

เก็บรหัสอักษร ห (รหัส 200) ไว้ที่คีย์ s (รหัส 115)

เก็บรหัสอักษร ะ (รหัส 204) ไว้ที่คีย์ t (รหัส 116)

เราจะกำหนดให้เป็นคอนสแตนต์ไว้ในโปรแกรมได้ดังนี้

```
CONST
   ThaiChar : ARRAY [#114..#116] OF Byte = (188,200,204)
```

ซึ่งมีความหมายดังนี้

ThaiChar [#114] มีค่าเป็น 188 ซึ่งเป็นรหัสอักษร พ

ThaiChar [#115] มีค่าเป็น 200 ซึ่งเป็นรหัสอักษร ห

ThaiChar [#116] มีค่าเป็น 204 ซึ่งเป็นรหัสอักษร ะ

ถ้ามีสเตตเมนต์

```
Write (Chr(ThaiChar [114]))
```

เมื่อโปรแกรมทำงานจะปรากฏอักษร พ ที่จอภาพ และสเตตเมนต์

```
Wrtie (Chr (114))
```

จะแสดงอักษร r ที่จอภาพ

ค่า 114 ซึ่งเป็นอินเด็กซ์ของอาร์เรย์ในตัวอย่างนี้ สำหรับในโปรแกรมจะเป็นค่าที่ได้รับจากการกดคีย์ ซึ่งอาจจะเขียนสเตตเมนต์ได้ดังนี้

```
Ch  :=  ReadKey ;
Write (Chr(Ch)) ;            { r }
Write (Chr(ThaiChar[ch]) { พ }
```

เมื่อโปรแกรมทำงานและคีย์ r ถูกกด ที่จอภาพจะปรากฏอักษร r และ พ ตามลำดับ

จะเห็นได้ว่าการที่โปรแกรมแสดงอักษรอังกฤษ หรืออักษรไทยย่อมขึ้นอยู่กับสเตตเมนต์ที่ใช้ ดังนั้นโปรแกรมจึงต้องมีเงื่อนไขสำหรับเลือกว่ากรณีใดแสดงอักษรอังกฤษ และกรณีใดแสดงอักษรไทย วิธีดำเนินการเกี่ยวกับเรื่องนี้ก็คือเราจะเลือกคีย์สำหรับเปลี่ยนอักษรไทย-อังกฤษขึ้นมา 1 คีย์ สมมติว่าเราเลือกคีย์ F10 เมื่อคีย์นี้ถูกกดโปรแกรมจะรับรหัสจากคีย์แล้วนำไปสร้างเงื่อนไขเลือกอักษร เช่น ถ้าเดิมเป็นอักษรอังกฤษ โปรแกรมจะเปลี่ยนให้เป็นอักษรไทย แต่ถ้าเดิมเป็นอักษรไทยโปรแกรมจะเปลี่ยนให้เป็นอักษรอังกฤษกลับไปกลับมาเช่นนี้ตลอดเวลา แสดงตัวอย่างสเตตเมนต์ดังต่อไปนี้

```
Thai := True;
REPEAT
   Ch := ReadKey;
   IF Ch = #0 THEN
   BEGIN
      Ch := ReadKey;
      IF Ch = F10 THEN Thai := NOT Thai;
   END;
   IF Ch > #31 THEN
   BEGIN
      IF Thai THEN DisplayChar (X,Y, ThaiChar[Ch]) {Thai}
              ELSE DisplayChar (X,Y, Ord(Ch));      {English}
      Inc (X,9); {next letter}
      IF X > GetMaxX - 9 THEN BEGIN X := 0; Inc (Y,15) END;
       {next row}
   END;
UNTIL Ch = Esc
```

การทำงานของสเตตเมนต์ชุดนี้ เริ่มด้วยการกำหนดคีย์ให้เป็นอักษรไทยต่อจากนั้นจะเป็นลูป REPEAT รอรับการกดคีย์

ถ้าคีย์อักษรถูกกด จะปรากฏอักษรไทยที่จอภาพ

ถ้าคีย์ F10 ถูกกด คีย์บอร์ดจะถูกเปลี่ยนเป็นภาษาอังกฤษ

เมื่อกดคีย์อักษรจะปรากฏอักษรอังกฤษที่จอภาพ

และถ้ากดคีย์ F10 อีก คีย์บอร์ดจะถูกเปลี่ยนเป็นภาษาไทย

การกดคีย์ F10 แต่ละครั้งจะมีผลให้คีย์บอร์ดถูกเปลี่ยนเป็นภาษาไทย และอังกฤษสลับกันไป ลูปจะสิ้นสุดการทำงานเมื่อกดคีย์ Esc

10.9.4 โปรแกรมแสดงอักษรกราฟิกส์ไทย-อังกฤษจากคีย์บอร์ด โปรแกรมแสดงอักษรกราฟิกส์ไทย-อังกฤษจากคีย์บอร์ดที่นำมาเป็นตัวอย่าง ในที่นี้มี 2 โปรแกรม แต่ละโปรแกรมปรับปรุงมาจากโปรแกรมแสดงอักษรจากฟอนต์สำหรับจอภาพ ได้แก่ โปรแกรมแสดงฟอนต์จาก RW และจากไฟล์อักษร 814.FON

หลักการสำคัญของการปรับปรุงโปรแกรมดังกล่าว ก็คือ การเพิ่มโพรซีเยอร์สำหรับทำหน้าที่เปลี่ยนรหัสอักษรไทย-อังกฤษ โดยในโพรซีเยอร์จะมีอาร์เรย์เก็บรหัสอักษรไทยเรียงตามลำดับอักษรอังกฤษที่อยู่บนคีย์เดียวกัน ตั้งแต่รหัส 32 ถึง 127 ดังต่อไปนี้

```
PROCEDURE ThaiCharKU(X,Y : Integer);
CONST ThaiChar : ARRAY[#32..#127] OF Byte =
        ( 32, 35, 46,132,133,134,235,165,136,137,135,139,191,
         162,211,187,166,195, 45, 47,190,180,215,219,163,179,
         169,196,176,168,201, 38,131,194, 37,167,173,172,210,
         170,223,177,227,198,197, 63,228,214,171,130,175,164,
```

```
        182,226,203, 34, 41,222, 40,184, 92,195,216,138, 96,
        189,217,209,161,207,178,208,225,193,224,206,199,181,
        220,183,192,213,188,200,204,218,202,212,185,221,186,
        174,124,44,126,127);
   F10 = #68;  {Toggle Thai-English}
   Esc = #27;  {Stop readkey}
```

จะเห็นได้ว่า มีการจัดลำดับรหัสอักษรเหมือนกับที่แสดงอยู่ในตารางที่ 10.6 และได้กำหนดคีย์ F10 สำหรับสลับอักษรไทย-อังกฤษ และคีย์ Esc สำหรับเลิกงาน

ตัวอย่าง โปรแกรม 10.13 ใช้สำหรับแสดงอักษรไทย-อังกฤษ ซึ่งได้จากการกดคีย์บนคีย์บอร์ด โปรแกรมจะเริ่มด้วยการให้ป้อนชื่อไฟล์ ซึ่งเป็นไฟล์เก็บอักษรขนาด 8 x 14 (เช่น ไฟล์ 814.FON) ต่อจากนั้น โปรแกรมจะอ่านอักษรจากไฟล์มาเก็บไว้ในอาร์เรย์แล้วคอยรับการกดคีย์ เราสามารถพิมพ์อักษรไทยและอังกฤษได้โดยการกดคีย์ F10 เพื่อเปลี่ยนชนิดของอักษร โปรแกรมจะเลิกทำงานเมื่อเรากดคีย์ Esc

โปรแกรม 10.13

```
PROGRAM KeyboardFont814;
USES Crt,Graph;
TYPE CharByte       = ARRAY[1..14] OF Byte;
     RecordOfChar   = RECORD
                        OneChar : CharByte;
                      END;
VAR GraphDriver,GraphMode : Integer;
    FontFile   : File OF RecordOfChar;
```

```
    FileName    : STRING[14];
    CharRec     : RecordOfChar;
    Ascii       : ARRAY [32..255] OF CharByte;
    Row,Col     : Integer;
    CharNo      : Byte;

PROCEDURE ReadFont;
VAR CharNo : Byte;
begin
   Assign(FontFile,FileName);
   Reset(FontFile);
   CharNo := 32;
   WHILE NOT Eof(FontFile) DO
   BEGIN
      Read(FontFile,CharRec);
      Ascii[CharNo] := CharRec.OneChar;
      Inc(CharNo);
   END;
   Close(FontFile);
end;

PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
BEGIN
  FOR ByteNo :=1 TO 14 DO
  BEGIN
     SetLineStyle(UserBitLn,Ascii[CharNo,ByteNo],1);
     IF GraphDriver {VGA} <> 7 THEN Line(X,Y X-15,Y) ELSE {Herc}
     Line(X,Y, X+15,Y);
     Inc(Y,1);
  END;
```

```
END;
PROCEDURE ThaiCharKU(X,Y : Integer);
CONST ThaiChar : ARRAY[#32..#127] OF Byte =
                    ( 32, 35, 46,132,133,134,235,165,136,137,135,139,191,
                     162,211,187,166,195, 45, 47,190,180,215,219,163,179,
                     169,196,176,168,201, 38,131,194, 37,167,173,172,210,
                     170,223,177,227,198,197, 63,228,214,171,130,175,164,
                     182,226,203, 34, 41,222, 40,184, 92,195,216,138, 96,
                     189,217,209,161,207,178,208,225,193,224,206,199,181,
                     220,183,192,213,188,200,204,218,202,212,185,221,186,
                     174,124,44,126,127);
      F10 = #68;  {Toggle Thai-English}
      Esc = #27;  {Stop readkey}
VAR Ch      : Char;
    Thai    : Boolean;
BEGIN
   Thai := True;
   REPEAT
      Ch := ReadKey;
      IF Ch = #0 THEN
      BEGIN
         Ch := ReadKey;
         IF Ch = F10 THEN Thai := NOT Thai;
         Ch := #0;
      END;
      IF Ch > #31 THEN
      BEGIN
         IF Thai THEN DisplayChar(X,Y,ThaiChar[Ch])  {Thai}
                 ELSE DisplayChar(X,Y,Ord(Ch));       {English}
```

```
      Inc(X,9);  {next letter}
      IF X > GetMaxX - 9 THEN BEGIN X := 0;Inc(Y,15) END; {next row}
    END;
  UNTIL Ch = Esc;
END;

BEGIN {Main program}
  WriteLn('Input Thai-English font from keyboard : FONT SIZE 8x14');
  Write('Enter 814.FON or other 8x14 font : ');
  ReadLn(FileName);
  ReadFont;
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  ClearDevice;
  OutTextXY(0,0,'Type any letter. Press F10 for THAI-ENGLISH keys.');
  OutTextXY(0,10,'                 Press Esc for QUIT program.');
  ThaiCharKU(0,30);
  CloseGraph
END.
```

ตัวอย่าง โปรแกรม 10.14 ใช้สำหรับแสดงอักษรไทย-อังกฤษ ซึ่งได้จากการกดคีย์บนคีย์บอร์ด โปรแกรมจะเริ่มด้วยการให้ป้อนชื่อไฟล์ ซึ่งเป็นไฟล์เก็บอักษรขนาด 8 x 24 (เช่น ไฟล์ FONT1.CSL และ FONT2.CSL จากโปรแกรม RW) เมื่อรับชื่อไฟล์แล้ว โปรแกรมจะอ่านอักษรจากไฟล์มาเก็บไว้ในอาร์เรย์และคอยรับการกดคีย์เราสามารถพิมพ์อักษรไทยและอังกฤษได้โดยการกดคีย์ F10 เพื่อเปลี่ยนชนิดของอักษร โปรแกรมจะเลิกทำงานเมื่อเรากดคีย์ Esc

โปรแกรม 10.14

```
PROGRAM KeyboardFont824;
USES Crt,Graph;
TYPE CharByte          = ARRAY[1..24] OF Byte;
     RecordOfChar  = RECORD
                          OneChar : CharByte;
                        END;
VAR GraphDriver,GraphMode : Integer;
    FontFile   : File OF RecordOfChar;
    FileName   : STRING[14];
    CharRec    : RecordOfChar;
    Ascii      : ARRAY [0..255] OF CharByte;
    Row,Col    : Integer;
    CharNo     : Byte;

PROCEDURE ReadFont;
VAR CharNo : Byte;
begin
   Assign(FontFile,FileName);
   Reset(FontFile);
   CharNo := 0;
   WHILE NOT Eof(FontFile) DO
   BEGIN
      Read(FontFile,CharRec);
      Ascii[CharNo] := CharRec.OneChar;
      Inc(CharNo);
   END;
   Close(FontFile);
end;
```

```
PROCEDURE DisplayChar(X,Y:Integer; CharNo:Byte);
VAR ByteNo : Byte;
BEGIN
  FOR ByteNo :=1 TO 24 DO
  BEGIN
    SetLineStyle(UserBitLn,Ascii[CharNo,ByteNo],1);
    IF GraphDriver {VGA} <> 7 THEN Line(X,Y X-15,Y) ELSE {Herc}
    Line(X,Y, X+15,Y);
    Inc(Y,1);
  END;
END;

PROCEDURE ThaiCharKU(X,Y : Integer);
CONST ThaiChar : ARRAY[#32..#127] OF Byte =
               ( 32, 35, 46,132,133,134,235,165,136,137,135,139,191,
                162,211,187,166,195, 45, 47,190,180,215,219,163,179,
                169,196,176,168,201, 38,131,194, 37,167,173,172,210,
                170,223,177,227,198,197, 63,228,214,171,130,175,164,
                182,226,203, 34, 41,222, 40,184, 92,195,216,138, 96,
                189,217,209,161,207,178,208,225,193,224,206,199,181,
                220,183,192,213,188,200,204,218,202,212,185,221,186,
                174,124,44,126,127);
     F10 = #68;  {Toggle Thai-English}
     Esc = #27;  {Stop readkey}
VAR Ch      : Char;
   Thai     : Boolean;
BEGIN
  Thai := True;
  REPEAT
    Ch := ReadKey;
```

```
      IF Ch = #0 THEN
      BEGIN
         Ch := ReadKey;
         IF Ch = F10 THEN Thai := NOT Thai;
         Ch := #0;
      END;
      IF Ch > #31 THEN
      BEGIN
         IF Thai THEN DisplayChar(X,Y,ThaiChar[Ch])   {Thai}
                 ELSE DisplayChar(X,Y,Ord(Ch));        {English}
         Inc(X,9);   {next letter}
         IF X > GetMaxX - 9 THEN BEGIN X := 0; Inc(Y,15) END; {next row}
      END;
   UNTIL Ch = Esc;
END;

BEGIN {Main program}
   WriteLn('Input Thai-English font from keyboard : FONT SIZE 8x24');
   Write('Enter FONT1.CSL, FONT2.CSL or other 8x24 font : ');
   ReadLn(FileName);
   ReadFont;
   GraphDriver := Detect;
   InitGraph(GraphDriver, GraphMode, '');
   ClearDevice;
   OutTextXY(0, 0,'Type any letter. Press F10 for THAI-ENGLISH keys.');
   OutTextXY(0,10,'                    Press Esc for QUIT program.');
   ThaiCharKU(0,30);
   CloseGraph
END.
```

โปรดสังเกตว่า โปรแกรมตัวอย่างที่นำมาแสดงในที่นี้จะใช้ได้ถูกต้องกับอักษรไทยรหัสเกษตรเท่านั้น ถ้าใช้กับอักษรไทยรหัสอื่นจะได้ผลไม่ถูกต้องเนื่องจากรหัสไม่ตรงกัน การปรับปรุงโปรแกรมเพื่อใช้กับภาษาไทยรหัสอื่นมีวิธีการดังนี้

1. เปลี่ยนรหัสอักษรไทยในตารางที่ 10.6 ให้ตรงกับรหัสภาษาไทยที่ต้องการจะใช้

2. เปลี่ยนค่ารหัสในอาร์เรย์ ThaiChar ให้ตรงกับรหัสภาษาไทยที่ต้องการจะใช้

11
โปรแกรมเกมคอมพิวเตอร์

ผู้ที่ได้เกี่ยวข้องกับคอมพิวเตอร์คงจะไม่มีใครที่ไม่รู้จักเกมคอมพิวเตอร์ และคงมีส่วนน้อยที่ไม่เคยเล่นเกมคอมพิวเตอร์เลย เกมคอมพิวเตอร์จะให้ความสนุกแก่ผู้เล่นและท้าทายผู้เล่นให้นำความสามารถทางด้านต่างๆ ออกมาใช้ เช่น ความสามารถทางด้านความเร็ว การกะระยะทาง การคำนวณ การเปรียบเทียบรูปทรง การฟังเสียง และการใช้ไหวพริบเพื่อแก้ปัญหาต่างๆ การเล่นเกมทำให้ผู้เล่นเกิดความสนุกในระดับหนึ่ง แต่ก็ยังสนุกน้อยกว่าการเขียนโปรแกรมเกมให้ผู้อื่นนำไปเล่น ผู้เขียนโปรแกรมเกมจะรู้สึกสนุกกับการตั้งกติกาของเกม การสร้างตัวละคร การดำเนินเรื่อง การให้กำลังใจและรางวัล รวมทั้งการแทรกมุขตลกหรือสิ่งที่ไม่คาดฝันให้เกิดขึ้นเพื่อเร้าความสนใจของผู้เล่น ในบทนี้เราจะได้ศึกษาเกี่ยวกับการเขียนโปรแกรมเกมคอมพิวเตอร์

11.1 ความหมายของเกมคอมพิวเตอร์

เกมคอมพิวเตอร์หมายถึง โปรแกรมสำเร็จรูปประเภทเกมที่เล่นด้วยเครื่องคอมพิวเตอร์ เกมสำหรับเครื่องไมโครคอมพิวเตอร์มักจะบันทึกอยู่ในดิสก์และมีคู่มือแนะนำการเล่นเกมแนบมาด้วย

11.2 อุปกรณ์สำหรับเล่นเกม

อุปกรณ์หลักที่ใช้สำหรับเล่นเกมคอมพิวเตอร์ได้แก่ เครื่องคอมพิวเตอร์ จอภาพและคีย์บอร์ด นอกจากนี้ ยังมีอุปกรณ์เสริมอีกหลายอย่างที่จะช่วยให้เล่นเกมได้สะดวกและสนุก เช่น จอยสติก (joystick) เมาส์ (mouse) แป้นกด (game paddle) และตารางกราฟิกส์ (graphics tablet) ตามปกติเกมส่วนใหญ่จะเล่นกับคีย์บอร์ดได้ แต่มีบางเกมที่ต้องการให้เล่นได้คล้ายของจริงมากที่สุด ผู้เขียนโปรแกรมจึงเพิ่มเติมคำสั่งให้สามารถใช้กับอุปกรณ์เสริมดังกล่าวได้

11.3 ประเภทของการเล่นเกม

เกมคอมพิวเตอร์แบ่งการเล่นออกเป็น 3 ประเภทคือ

11.3.1 เล่นคนเดียว หมายถึงเกมที่ผู้เล่นเป็นผู้แก้ปัญหาของเกมนั้นด้วยตัวเอง โดยที่คอมพิวเตอร์ไม่ได้มีส่วนในการจัดการกับเกมนั้นเลย เช่น เกมพายเรือในคลองคด ซึ่งผู้เล่นจะต้องใช้ความสามารถควบคุมเรือจากจุดเริ่มต้นให้ไปถึงที่หมายโดยให้เรือกระทบกับฝั่งน้อยครั้งที่สุด จะเห็นได้ว่าเกมประเภทนี้คอมพิวเตอร์ไม่ได้เข้ามามีส่วนร่วมในการเล่น เพียงแต่ทำหน้าที่เป็นเครื่องมือสำหรับเล่นเกมเท่านั้น

11.3.2 เล่น 2 คน หมายถึง ผู้เล่นเกมมี 2 คน แต่ละคนจะเล่นเพื่อให้ชนะคู่ต่อสู้ เช่น เกมชกมวย ซึ่งผู้เล่นแต่ละคนต่างพยายามชกให้คู่ต่อสู้แพ้ สำหรับเกมประเภทนี้จะเห็นได้ว่า ผู้เล่นเกมเป็นคนทั้งคู่ คอมพิวเตอร์ทำหน้าที่เป็นเครื่องมือเล่นเกมเหมือนกับเกมประเภทแรกที่ได้กล่าวมาแล้ว

11.3.3 เล่นกับคอมพิวเตอร์ หมายถึง ผู้เล่นเกมฝ่ายหนึ่งเป็นคนแต่อีกฝ่ายหนึ่งเป็นเครื่องคอมพิวเตอร์ โดยฝ่ายที่เป็นคนอาจจะมีคนเดียวหรือ 2 คนก็ได้ เช่น เกมยิงศัตรูต่างดาว ซึ่งคอมพิวเตอร์จะทำหน้าที่สร้างมนุษย์ต่างดาวที่มีความสามารถต่างๆ ออกมาต่อสู้กับคน ฝ่ายคนก็จะต้องคอยยิงต่อสู้เพื่อป้องกันไม่ให้มนุษย์ต่างดาวเข้ามาในโลก ในกรณีเช่นนี้จะเห็นได้ว่า คอมพิวเตอร์ได้เข้ามามีบทบาททั้งในฐานะที่เป็นเครื่องมือเล่นเกมและเป็นผู้เล่นเกมด้วย

11.4 ส่วนประกอบของเกมคอมพิวเตอร์

ส่วนประกอบที่เป็นพื้นฐานของเกมคอมพิวเตอร์ซึ่งทุกๆ เกมจะต้องมีก็คือ กติกาของเกมและวิธีการเล่นเกม ซึ่งมีรายละเอียดดังนี้

11.4.1 กติกาของเกม เป็นกฎเกณฑ์และเงื่อนไขต่างๆ ของเกม ที่ผู้เล่นเกมจะต้องทราบและถือปฏิบัติตาม เช่น เกมบวกเลข มีกติกาว่าเมื่อคอมพิวเตอร์แสดงเลข 2 จำนวนที่จอภาพ ให้ผู้เล่นเกมป้อนผลบวกของเลขคู่นั้นเข้าคอมพิวเตอร์ ถ้าตอบถูกตั้งแต่ครั้งแรก คอมพิวเตอร์จะให้คะแนนเป็น 6 ถ้าตอบไม่ถูก ผู้เล่นจะต้องป้อนคำตอบใหม่ ถ้าตอบถูกในครั้งนี้จะได้ 5 คะแนน ถ้ายังตอบไม่ถูกก็ต้องป้อนคำตอบเข้าไปใหม่ ทำเช่นนี้เรื่อยๆ โดยคอมพิวเตอร์จะยอมให้ป้อนคำตอบไม่เกิน 6 ครั้ง ถ้าตอบถูกในครั้งหลัง คะแนนจะลดลงจากการตอบถูกในครั้งก่อนครั้งละ 1 คะแนน เมื่อตอบถูกแล้วคอมพิวเตอร์จะเริ่มคำถามในข้อต่อไป ถ้าตอบไม่ถูกเลยทั้ง 6 ครั้งจะได้คะแนน 0 และคอมพิวเตอร์จะเริ่มคำถามในข้อต่อไป ผู้เล่นจะเลิกโปรแกรมเมื่อกดคีย์ Esc ตอนที่คอมพิวเตอร์ถามว่าจะเริ่มโจทย์ข้อต่อไปหรือจะหยุดเล่น

ถ้าต้องการเล่นเกมให้สนุกและมีโอกาสชนะมากกว่าแพ้ ผู้เล่นเกมจะต้องศึกษาและทำความเข้าใจกับกติกาของเกมให้ชัดเจน ตามปกติเกมแต่ละเกมจะมีคู่มือมาด้วย ซึ่งผู้เขียนเกมอาจจะทำเป็นเอกสารแนบมากับดิสก์หรือบันทึกไว้ในดิสก์เลยก็ได้

11.4.2 วิธีการเล่นเกม เป็นการกำหนดวิธีเล่นเกมตามกติกาที่ตกลงกันไว้ โดยใช้คอมพิวเตอร์เป็นเครื่องมือ ซึ่งจะมีการกำหนดรายละเอียดต่างๆไว้อย่างครบถ้วน เช่น ถ้าในกติกาของเกมระบุไว้ว่าให้มีการยิงปืน ในส่วนของวิธีการเล่นเกมก็จะต้องกำหนดให้แน่นอนว่าจะยิงด้วยการกดคีย์ใด ถึงขั้นนี้เราจะเห็นได้ว่าในเกมที่มีกติกาเหมือนกัน เวลาเล่นอาจจะต้องมีวิธีแตกต่างกัน เช่น บางเกมกำหนดให้ยิงโดยการกดคีย์วรรค บางเกมให้กดคีย์ Enter ดังนั้น ถ้าต้องการเล่นเกมให้ได้คล่องแคล่วและครบถ้วน เราจะต้องศึกษาวิธีการเล่นเกมนั้นให้เข้าใจอย่างถ่องแท้และครบถ้วนตามที่ผู้เขียนโปรแกรมได้กำหนดเอาไว้

นอกจากส่วนประกอบพื้นฐานดังกล่าวนี้แล้ว อาจจะมีส่วนประกอบอื่นๆ เพิ่มเติมอีกก็ได้เช่น เสียงประกอบเกม คำสั่งสำหรับเล่นเกมกับอุปกรณ์เสริม ไพรซีเยอร์ตรวจรหัสผู้มีสิทธิเล่นเกมซึ่งจะมีในโปรแกรมชนิดที่ห้ามทำการก๊อบปี้

11.5 วิธีการสร้างเกมคอมพิวเตอร์

การสร้างเกมคอมพิวเตอร์ หมายถึง การสร้างกติกา สร้างเรื่องราว กำหนดกฎเกณฑ์ เงื่อนไขและวิธีการเล่นให้ผู้เล่นปฏิบัติตาม ซึ่งจะเห็นได้ว่าผู้เขียนโปรแกรมเกมเป็นผู้ที่มีอิทธิพลเหนือผู้เล่น โดยเฉพาะอย่างยิ่งสำหรับเกมที่มีผู้นิยมเล่นกันมาก อิทธิพลของผู้เขียนโปรแกรมเกมจะแผ่ครอบคลุมบริเวณกว้าง อาจจะทั่วประเทศหรือทั่วโลก ดังเช่นที่เราได้เคยทราบข่าวของการนิยมชมชอบต่อเกมบางเกมของคนทั่วโลกมาแล้ว ด้วยเหตุนี้จึงอาจกล่าวได้ว่า การเล่นเกมทำให้เกิดความสนุกแต่การเขียนเกมมีความสนุกยิ่งกว่า ในหัวข้อนี้จะได้กล่าวถึงวิธีการสร้างเกมคอมพิวเตอร์ ซึ่งมีรายละเอียดดังต่อไปนี้

11.5.1 กำหนดกติกาของเกม การสร้างเกมคอมพิวเตอร์จะต้องเริ่มด้วยการกำหนดกติกาการเล่นเกมขึ้นมาก่อน ซึ่งในขั้นต้นอาจจะกำหนดไว้อย่างคร่าวๆ เช่น เจ้าหญิงแห่งนคร XYZ ถูกลักพาตัวไปกักขังไว้ในถ้ำลึกลับแห่งหนึ่ง ผู้เล่นเกมจะต้องติดตามเพื่อช่วยเหลือเจ้าหญิงกลับคืนมาให้ทันภายในเวลาที่กำหนด เมื่อได้กติกาหลักเช่นนี้แล้วจึงค่อยเพิ่มเติมรายละเอียดต่างๆเข้าภายหลัง เช่น เพิ่มเติมเส้นทางการเดินทาง กติกาการต่อสู้กับผู้ขัดขวางการเดินทาง ความสามารถของอาวุธต่างๆ ที่ใช้ เทคนิคแผนการและท่าทางของการต่อสู้

11.5.2 กำหนดเนื้อเรื่อง เพื่อให้เกมสนุกสนานอาจจะต้องสร้างเกมให้เป็นเรื่องราวต่อเนื่องกัน ซึ่งจะต้องมีการสร้างฉาก สร้างเสียงประกอบและสร้างตัวละครขึ้นมาจำนวนหนึ่ง สำหรับเกมอย่างง่ายข้อนี้อาจไม่จำเป็นต้องมีก็ได้

11.5.3 กำหนดวิธีการเล่น เป็นการกำหนดว่าเกมจะเริ่มอย่างไร จะต้องใช้คีย์ไหนเพื่อทำอะไร ถ้าจะเลิกเล่นหรือพักชั่วคราวจะต้องทำอย่างไร

11.5.4 กำหนดอุปกรณ์ที่ใช้เล่นเกม เช่น กำหนดชนิดของคอมพิวเตอร์ว่าเป็นคอมพิวเตอร์แบบไหน รุ่นไหน มีหน่วยความจำเท่าไร ต้องใช้ฮาร์ดดิสก์หรือไม่ จอภาพเป็นแบบใด ต้องใช้อุปกรณ์เสริมหรือไม่

เมื่อได้กำหนดรายละเอียดต่างๆ ไว้เรียบร้อยแล้วจึงเริ่มลงมือทำการเขียนโปรแกรม การเขียนโปรแกรมเกมหนึ่งๆ อาจจะต้องใช้ผู้เขียนหลายคนเพื่อช่วยกันเขียนในส่วนต่างๆ เช่น เขียนฉาก เขียนตัวละคร เขียนเนื้อเรื่อง และสร้างเสียงประกอบเกม

11.6 ภาษาคอมพิวเตอร์ที่ใช้สร้างเกม

กล่าวได้ว่าภาษาคอมพิวเตอร์ทุกภาษา สามารถนำมาเขียนเป็นโปรแกรมเกมได้ แต่อาจจะได้ผลไม่เท่ากัน โดยเฉพาะทางด้านเทคนิคพิเศษบางอย่างที่ภาษาระดับสูงบางภาษาหรือบางรุ่นไม่สามารถจัดการได้ ดังนั้นในการสร้างเกมที่มีเทคนิคพิเศษมากๆ จึงจำเป็นต้องใช้ภาษาระดับกลาง เช่น ภาษาแอสเซมบลี (assambly) หรือภาษาระดับสูงที่สามารถเชื่อมโยงกับภาษาระดับต่ำกว่าได้

เทอร์โบปาสคาล เป็นภาษาระดับสูงที่สามารถเชื่อมโยงกับภาษาแอสเซมบลีได้ จึงนำมาใช้เขียนโปรแกรมเกมได้ดีพอสมควร อย่างไรก็ตาม โปรแกรมเกมที่นำมาแสดงเป็นตัวอย่างในที่นี้ทั้งหมด เขียนขึ้นโดยใช้เฉพาะส่วนที่เป็นภาษาสูงของเทอร์โบปาสคาลเท่านั้น

11.7 ตัวอย่างโปรแกรมเกม

เพื่อให้เห็นแนวทางของการสร้างเกมคอมพิวเตอร์ จึงได้นำโปรแกรมเกมมาแสดงเป็นตัวอย่างไว้ในที่นี้รวม 8 เกม ทุกเกมมีเสียงประกอบเพื่อเร้าความสนใจและทุกเกมเล่นด้วยคีย์บอร์ดเพียงอย่างเดียว บางเกมเป็นโหมดเท็กซ์และบางเกมเป็นโหมดกราฟิกส์แต่ละเกมจะมีความยากง่ายในการสร้างแตกต่างกัน ดังต่อไปนี้

11.7.1 เกมบวกเลข

1. โหมดของจอภาพ เท็กซ์

2. กติกา คอมพิวเตอร์จะสร้างเลขสุ่มขึ้นมา 2 จำนวน แต่ละจำนวนเป็นเลขไม่เกิน 2 หลัก (คือมีค่าตั้งแต่ 0 ถึง 99) และแสดงเลข 2 จำนวนนั้นที่จอภาพให้ผู้เล่นเกมหาผลบวกของเลข 2 จำนวนนั้น แล้วป้อนคำตอบเข้าคอมพิวเตอร์ ถ้าตอบถูกตั้งแต่ครั้งแรกคอมพิวเตอร์จะให้คะแนนเป็น 6 ถ้าตอบไม่ถูก ผู้เล่นจะต้องป้อนคำตอบใหม่ ถ้าตอบถูกในครั้งนี้จะได้ 5 คะแนน ถ้ายังตอบไม่ถูกก็ต้องป้อนคำตอบเข้าไปใหม่ ทำเช่นนี้ไปเรื่อยๆ โดยคอมพิวเตอร์จะยอมให้ป้อนคำตอบข้อละไม่เกิน 6 ครั้ง ถ้าตอบถูกในครั้งหลังคะแนนจะลดลงจากการตอบถูกในครั้งก่อนครั้งละ 1 คะแนน เมื่อตอบถูกแล้วคอมพิวเตอร์จะเริ่มคำถามในข้อต่อไป ถ้าตอบไม่ถูกเลยทั้ง 6 ครั้งจะได้คะแนน 0 และคอมพิวเตอร์จะเริ่มคำถามในข้อต่อไป

3. วิธีเล่น เมื่อเริ่มโปรแกรมที่จอภาพจะเป็นดังแสดงในรูปที่ 11.1

```
Total Item        : 0                    Total Score : 0

Number of Guess   : 0                    Item  Score :

                          Press Any Key to Start

                                    ESC  to Stop
```

รูปที่ 11.1 จอภาพเริ่มต้นของเกมบวกเลขซึ่งกำลังรอให้เลือกเริ่มหรือจบเกม

แต่ละรายการบนจอภาพมีความหมายดังนี้

Total Item หมายถึงจำนวนข้อทั้งหมดของคำถาม

Number of Guess หมายถึงจำนวนครั้งทั้งหมดที่ตอบ

ถ้า Number of Guess = Total Item แสดงว่าตอบถูกทุกครั้ง

ถ้า Number of Guess > Total Item แสดงว่ามีการตอบผิด ซึ่งจำนวนครั้งที่ตอบผิดจะเท่ากับจำนวนที่มากกว่า

Total Score หมายถึงคะแนนรวมทั้งหมดที่ทำได้

Item Score หมายถึงคะแนนที่จะได้จากข้อนั้น จากการตอบในครั้งนั้นคะแนนจะเริ่มจาก 6 ถ้าตอบไม่ถูกคะแนนจะลดลงครั้งละ 1 คะแนน

Press Any Key to Start หมายถึงให้กดคีย์เพื่อเริ่มคำถาม

ESC to Stop หมายถึงให้กดคีย์ Esc เพื่อเลิกเล่นเกม

4. ข้อสังเกต เนื่องจากเกมนี้ไม่ได้จับเวลาที่ใช้ในการหาคำตอบ ดังนั้นจึงควรคิดให้ถูกต้อง แล้วจึงค่อยตอบ ผลการเล่นจะดีที่สุดเมื่อ

Number of Guess = Total Item

5. โปรแกรม รายละเอียดของโปรแกรมเกมบวกเลข แสดงในโปรแกรม 11.1

โปรแกรม 11.1

```
PROGRAM AddTwoDigitNumber;
USES Crt;
VAR TotalSc,TotalItem,GuessNo : Integer;
    Item   : Char;

PROCEDURE Addition;
VAR Opt1,Opt2,Answer       : Real;
```

```
   Result,Score,Row,Col      : Integer;
BEGIN
   Score :=6;
   REPEAT
      Randomize;
      Opt1 := Random(100);
      Opt2 := Random(100);
   UNTIL (Opt1+opt2 < 100) AND ((Trunc(Opt1) MOD 10)+
        (Trunc(Opt2) MOD 10) < 10);
   Window(52,5,55,7); WriteLn(Score);
   Row := 0; Col :=2;
   REPEAT
      Row := Row+1;
      Window(30,7,50,12); GotoXy(Col,Row);
      Write(Opt1:2:0,' + ',Opt2:2:0,' = ');  ReadLn(Result);
      Answer := Opt1+Opt2;
      IF Result <> Answer THEN
      BEGIN
         Inc(GuessNo);
         Score := Score-1;
      END;
      IF Result = Answer THEN
      BEGIN
         Inc(GuessNo);
         Sound(300);Delay(50);
         Sound(900);Delay(50);Nosound
      END;
      Window(52,5,55,7); ClrScr; WriteLn(Score);
   UNTIL (Result = Answer) OR (Row = 6);
   Window(1,1,80,25);
```

```
  TotalSc := TotalSc + Score;
END;

BEGIN {MAIN}
  ClrScr;
  TotalSc := 0;
  TotalItem :=0;
  GuessNo := 0;
  GotoXY(6,2);  Write('Total Items    : ');
  GotoXY(6,3);  Write('Number of Guess : ');
  GotoXY(38,2); Write('Total Score : ');
  GotoXY(38,5); Write('Item  Score : ');
  REPEAT
    GotoXY(25,2); Write(TotalItem : 3);
    GotoXY(25,3); Write(GuessNo : 3);
    Window(52,2,60,3);WriteLn(TotalSc);
    Window(30,7,54,25); ClrScr;
    WriteLn('Press Any Key to Start ');
    Write('           ESC  to  Stop   ');
    Item := ReadKey ;
    IF Item <> #27 THEN
    BEGIN
      Sound(200); Delay(100);Nosound;
      Window(1,6,80,10);ClrScr;
      Addition;  Inc(TotalItem);
    END;
  UNTIL Item = #27;
END.
```

11.7.2 เกมหาตัวบวก

1. โหมดของจอภาพ เท็กซ์

2. กติกา เกมนี้มาจากคำถามที่ว่า 15 บวกกับอะไรจึงจะได้ 50 ดังนั้นเกมจึงเริ่มด้วยการให้ผู้เล่นป้อนเลขจำนวนแรกซึ่งเป็นตัวตั้งเข้าไป แล้วป้อนเลขจำนวนที่สองซึ่งเป็นผลบวก ต่อจากนั้นคอมพิวเตอร์จึงแสดงตัวบวกให้เห็น

3. วิธีเล่น เมื่อเริ่มโปรแกรมที่จอภาพจะเป็นดังแสดงในรูปที่ 11.2

```
              X     +     ?     =     Y

   Enter   X : _
```

รูปที่ 11.2 จอภาพแรกของเกมหาตัวบวกซึ่งกำลังรอการป้อนเลขตัวตั้ง

เมื่อป้อนเลขตัวตั้งเสร็จแล้ว โปรแกรมจะให้ป้อนคำตอบดังแสดงในรูปที่ 11.3

```
         X     +     ?     =     Y

        50     +     ?     =

                        Enter Y : _
```

รูปที่ 11.3 การรอรับเลขที่เป็นผลบวก

ต่อจากนั้นโปรแกรมจะแสดงเลขซึ่งเป็นตัวบวก ดังแสดงในรูปที่ 11.4

```
          X      +      ?      =      Y
          50     +      70     =      120


             Play again Y/N _
```

รูปที่ 11.4 ตัวบวกกะพริบซึ่งเป็นคำตอบและโปรแกรมรอการเลือกเล่นอีกหรือเลิก

4. ข้อสังเกต ถ้าเราป้อนตัวเลขที่เป็นผลบวกด้วยค่าที่ต่ำกว่าค่าตัวตั้งคำตอบที่ได้จะเป็นค่าลบ เช่น 15 + –5 = 10

5. โปรแกรม รายละเอียดของโปรแกรมเกมหาตัวบวกแสดงในโปรแกรม 11.2

โปรแกรม 11.2

```
PROGRAM WhatNumber;
USES Crt;
VAR First,Last,Result : Integer;
    Ch : Char;
BEGIN
   REPEAT
      ClrScr;
      WriteLn('            X      +      ?      =      Y');
      WriteLn('Enter X : ');
```

```
    GotoXY(11,2);Read(First);
    GotoXY(1,2); ClrEol;
    GotoXY(1,3); Write(First:12, '+':5, '?':5,'=':6, 'Y':6);
    GotoXY(28,4);Write('Enter Y : '); Read(Result);
    GotoXY(1,4); ClrEol;
    GotoXY(1,3); Write(First:12, '+':5);
    TextColor(White+128); {bright & blink}
    Write(Result-First:5);
    TextColor(Green);
    Write('=':6, Result:6);
    GotoXY(15,7); Write('Play again Y/N ');
    Ch := ReadKey;
  UNTIL Ch IN ['N','n'];
END.
```

11.7.3 เกมไพ่ 21 (blackjack)

เกมนี้ดัดแปลงมาจากเกม blackjack ในหนังสือ Hands-On Turbo Pascal เขียนโดย Larry Joel Goldstein

1. โหมดของจอภาพ เท็กซ์

2. กติกา เจ้ามือจะแจกไพ่ให้ตนเองและคู่เล่นคนละ 2 ใบ โดยไพ่ของคู่เล่นจะหงายขึ้นทั้งคู่ แต่ของเจ้ามือหงายเพียงใบเดียว ผู้ชนะคือ ผู้ที่มีแต้มรวมเท่ากับ 21 แต้มหรือใกล้เคียง 21 มากที่สุด หลังจากได้รับแจกไพ่แล้ว คู่เล่นอาจจะขอเพิ่มไพ่อีกก็ได้เพื่อเพิ่มแต้มให้มากขึ้น เมื่อคู่เล่นหยุดขอเพิ่มไพ่แล้ว เจ้ามือจะเปิดไพ่ของตนและอาจจะเพิ่มไพ่ขึ้นอีกจนเป็นที่พอใจ ผู้ที่มีแต้มเท่ากับ 21 หรือใกล้เคียงกว่าจะเป็นผู้ชนะ

แต้มของไพ่แต่ละใบจะเท่ากับตัวเลขที่ปรากฏอยู่บนไพ่ใบนั้น สำหรับไพ่ที่เป็น J Q และ K กำหนดให้เป็น 10 แต้ม ส่วนไพ่ที่เป็น A โดยทั่วไปจะกำหนดให้เป็น 1 แต้ม ยกเว้นกรณีตอนแจกไพ่ 2 ใบแรก ถ้ามีไพ่ 10 แต้มอยู่ 1 ใบและไพ่ A อีก 1 ใบให้ถือว่า A มีค่าเป็น 11 แต้ม และผลรวมจะเป็น 21 แต้มพอดี ผู้ถือไพ่คู่นี้เป็นผู้ชนะทันที

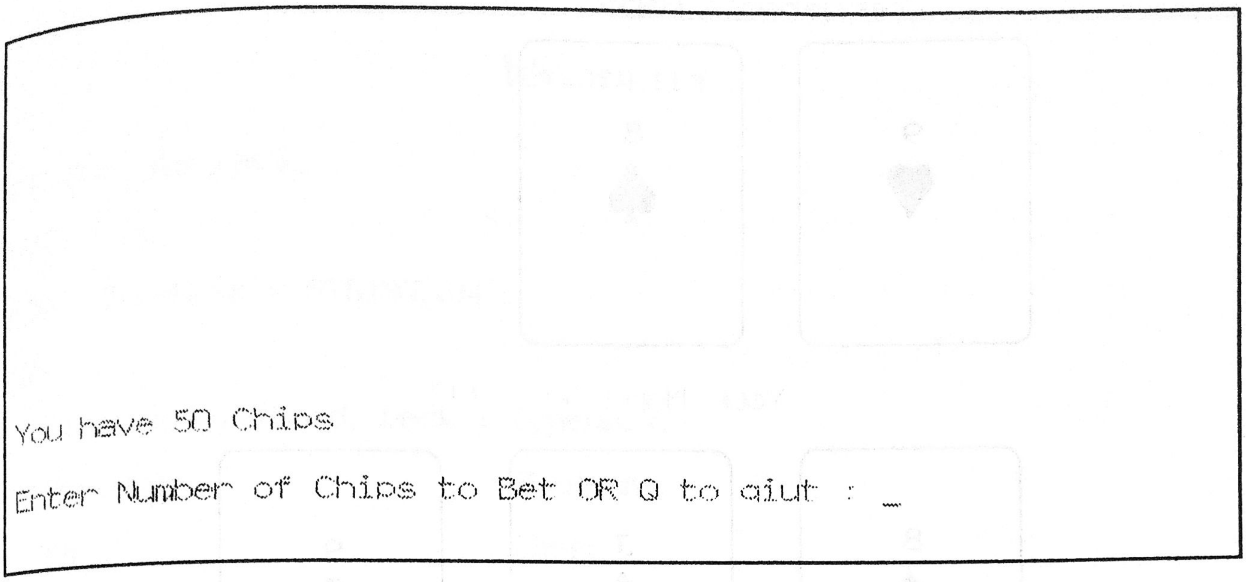
3. วิธีเล่น เกมนี้กำหนดให้คอมพิวเตอร์เป็นเจ้ามือ ก่อนเล่นผู้เล่นจะต้องนำเงินไปแลกเป็นคูปองมาก่อน คูปองทำด้วยพลาสติกมีลักษณะคล้ายเงินเหรียญ นิยมเรียกคูปองนี้ว่า ชิพ (chip) เมื่อเริ่มเกม คอมพิวเตอร์จะให้ป้อนจำนวนชิพที่เรามี ซึ่งจะแสดงจอภาพดังรูปที่ 11.5

```
Start Play BLACKJACK
Enter the Number of Chips (1..32767) to Start with : _
```

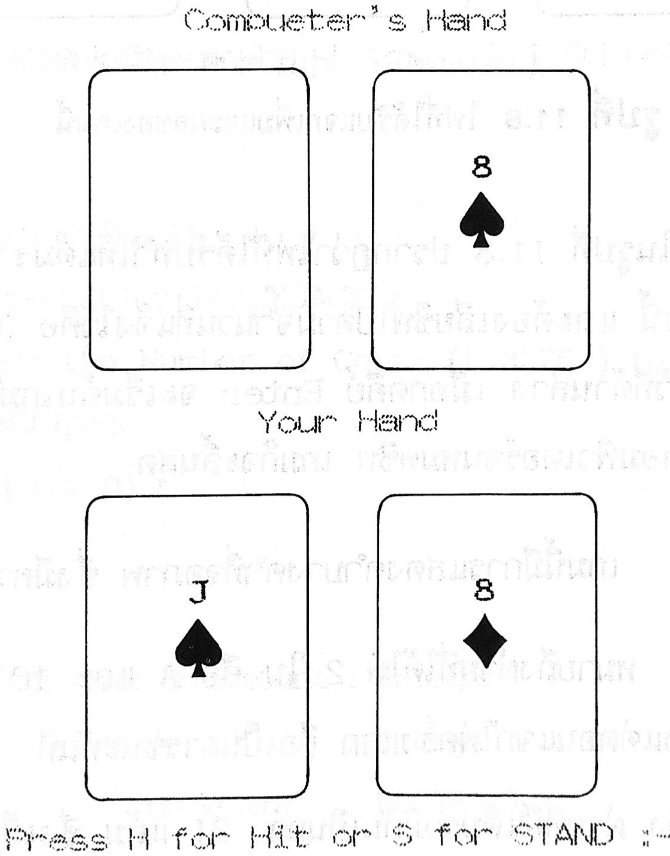
รูปที่ 11.5 จอภาพตอนเริ่มเกมไพ่ 21 ซึ่งเป็นการรอรับจำนวนชิพ

ต่อจากนั้นคอมพิวเตอร์จะแสดงจำนวนชิพที่เรามีอยู่ ซึ่งในที่นี้สมมติว่ามี 50 อัน และให้ป้อนจำนวนชิพที่เราจะเล่นในเกมนี้ ซึ่งจะแสดงจอภาพดังรูปที่ 11.6

เมื่อป้อนข้อมูลเรียบร้อยแล้ว คอมพิวเตอร์จะเริ่มแจกไพ่ให้ฝ่ายละ 2 ใบ โดยไพ่ของคอมพิวเตอร์จะคว่ำไว้ 1 ใบ และถามว่าต้องการจะรับไพ่เพิ่ม (hit) หรือพอแล้ว (stand) ดังตัวอย่างในรูปที่ 11.7

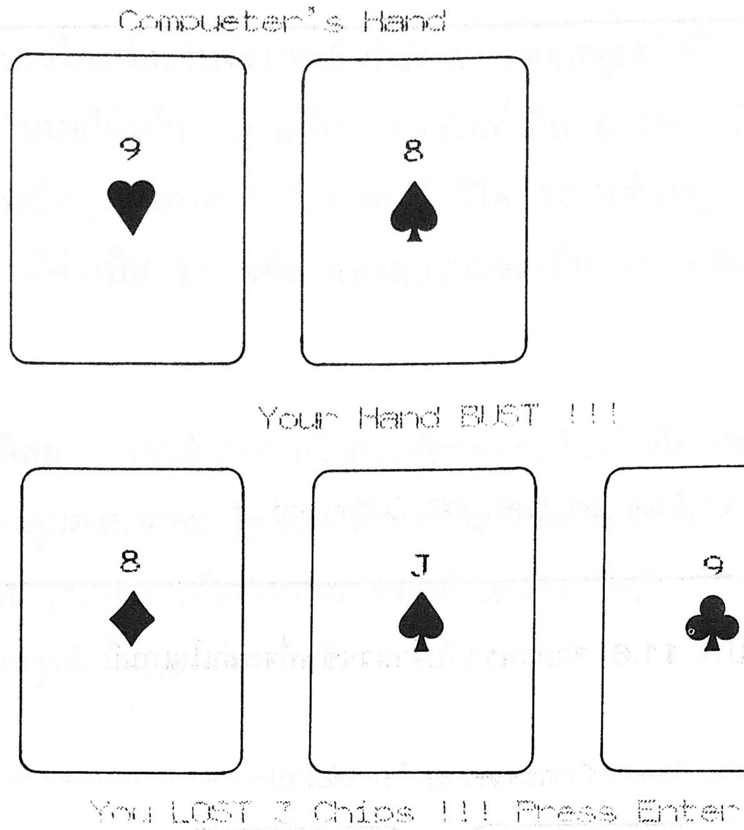


รูปที่ 11.6 จอภาพรอรับจำนวนชิพที่จะเล่นในเกมนี้



รูปที่ 11.7 ไพ่ตอนเริ่มแจกและถามว่าต้องการไพ่หรือพอแล้ว

ถ้าเรากดคีย์ H เพื่อขอไพ่เพิ่ม คอมพิวเตอร์จะแจกไพ่มาให้อีก 1 ใบดังแสดงในรูปที่ 11.8



รูปที่ 11.8 ไพ่ที่ได้รับแจกเพิ่มและผลของเกมนี้

ตามตัวอย่างในรูปที่ 11.8 ปรากฏว่าไพ่ที่ได้รับทำให้แต้มรวมมีค่าเกิน 21 ซึ่งแสดงว่าเราแพ้ในเกมนี้ และต้องเสียชิพไปตามจำนวนที่แจ้งไว้คือ 3 ชิพ ตามข้อความที่คอมพิวเตอร์แสดงไว้ที่ด้านล่าง เมื่อกดคีย์ Enter จะเริ่มต้นเกมใหม่เหมือนดังรูปที่ 11.6 และถ้าเราแพ้คอมพิวเตอร์จนหมดชิพ เกมก็จะสิ้นสุด

4. ข้อสังเกต เกมนี้มีการแสดงคำบางคำที่จอภาพ ซึ่งมีความหมายดังต่อไปนี้

BLACKJACK หมายถึงฝ่ายที่ได้ไพ่ 2 ใบ คือ A และ 10 J Q หรือ K ซึ่งรวมกันได้ 21 แต้มตั้งแต่ตอนแจกไพ่ครั้งแรก ถือเป็นการชนะทันที

BUST หมายถึง ฝ่ายที่มีแต้มรวมกันเกินกว่า 21 แต้ม ซึ่งเป็นฝ่ายแพ้

BANKRUPT หมายถึง ฝ่ายที่แพ้จนหมดชิพซึ่งไม่มีสิทธิจะเล่นอีกต่อไป

5. โปรแกรม รายละเอียดของโปรแกรมเกมไพ่ 21 แสดงในโปรแกรม 11.3

โปรแกรม 11.3

```
PROGRAM Blackjack;
USES Crt;
TYPE TypeDeck = STRING[104];
VAR
  ComputerFirstCard, Deck : TypeDeck;
  GameOver                : Boolean;
  Ch                      : Char;
  PlayerTotal, ComputerTotal, PlayerNumAce, ComputerNumAce,
  BetAmount, PlayerNumCard, ComputerNumCard, NumChips : Integer;

PROCEDURE GetNumChips(VAR NumChips : Integer);
BEGIN
  REPEAT
    ClrScr; GotoXY(1,23);
    WriteLn('Start Play BLACKJACK');
    Write('Enter the Number of Chips (1..32767) to Start with : ');
    ReadLn(NumChips)
  UNTIL (NumChips > 0)
END;

PROCEDURE PlaceBet (VAR BetAmount : Integer;
                    VAR NumChips  : Integer;
                        GameOver  : Boolean);
VAR
  ErrorCode    : Integer;
  ChoiceString : STRING[5];
BEGIN
  REPEAT
```

```
      ErrorCode := 0;
      GotoXY(1,23);ClrEol;
      WriteLn('You have ', NumChips,' Chips'); ClrEol;
      Write('Enter Number of Chips to Bet OR Q to quit : ');
      ReadLn(ChoiceString);
      IF ChoiceString[1] IN['Q','q'] THEN
      BEGIN
         GameOver  := True;
         NumChips  := 0;
         BetAmount := 1
      END
      ELSE Val (ChoiceString, BetAmount, ErrorCode);
   UNTIL ((BetAmount IN [1..NumChips]) AND (ErrorCode = 0)) OR GameOver
END;

PROCEDURE DrawCard(Card : TypeDeck; NumCard : Integer;
                   ComputerCard, CompFirstOver : Boolean);
CONST
   TopOfCard     = #213 + #205 + #205 + #205 + #205 +
                   #205 + #205 + #205 + #205 + #187;
   SideOfCard    = #179 + '            ' + #186;
   BottomOfCard = #192 + #196 + #196 + #196 + #196 +
                   #196 + #196 + #196 + #196 + #189;
VAR SideNum, XPos, Ypos : Integer;

  FUNCTION CardSymbol(CharSymbol : Char) : Char;
  BEGIN {CardSymbol}
    CASE CharSymbol of
      'H' : CardSymbol := #3;
      'D' : CardSymbol := #4;
```

```
      'C' : CardSymbol := #5;
      'S' : cardSymbol := #6
    END
  END;

BEGIN {DrawCard}
  XPos := 11 * (NumCard - 1) + 1;
  IF ComputerCard THEN YPos := 3
                  ELSE YPos := 14;
  GotoXY(XPos, YPos);
  Write(TopOfCard);
  FOR SideNum := 1 to 5 DO
  BEGIN
    GotoXY(XPos, Ypos + SideNum);
    Write(SideOfCard)
  END;
  GotoXY(XPos, YPos + 6);
  Write(BottomOfCard);
  GotoXY(XPos + 4, YPos + 2);
  IF NOT (ComputerCard AND (NumCard = 1)) OR CompFirstOver THEN
   BEGIN
     IF (Card[1] = '0') THEN Write('1');
     Write(Card[1]);
     GotoXY(XPos + 4, YPos + 3); Write(CardSymbol(Card[2]))
   END
END;

PROCEDURE DealACard(VAR ComputerFirstCard, Deck : TypeDeck;
                    VAR TotalValue, NumCard, NumAce : Integer;
                        ComputerCard : Boolean);
```

```
VAR Card : TypeDeck;

   FUNCTION CardScore(Card : Char) : Integer;
   BEGIN
      CASE Card of
         'A'                    : CardScore := 11;
         'J', 'Q', 'K', '0' : CardScore := 10;
         ELSE                     CardScore := Ord(Card) - 48
      END
   END;

BEGIN {DealACard}
   Card := Copy(Deck, 1, 2);
   Deck := Copy(Deck, 3, Length(Deck) - 2);
   TotalValue := TotalValue + CardScore(Card[1]);
   IF Card[1] = 'A' THEN NumAce := NumAce + 1;
   IF (TotalValue > 21) AND (NumAce > 0) THEN
   BEGIN
      TotalValue := TotalValue - 10;
      NumAce := NumAce - 1
   END;
   NumCard := NumCard + 1;
   IF ComputerCard AND (NumCard = 1) THEN ComputerFirstCard := Card;
   Sound(700); Delay(50); NoSound;
   DrawCard(Card, NumCard, ComputerCard, False)
END;

PROCEDURE SetupHand(VAR ComputerFirstCard, Deck : TypeDeck;
                    VAR PlayerTotal, ComputerTotal,
                        PlayerNumAce, ComputerNumAce,
```

```
            PlayerNumCard, ComputerNumCard : Integer);

PROCEDURE SetupDeck (VAR Deck : TypeDeck);
CONST
  FullDeck = 'AC2C3C4C5C6C7C8C9C0CJCQCKCAD2D3D4D5D6D7D8D9D0DJDQDKD'+
             'AH2H3H4H5H6H7H8H9H0HJHQHKHAS2S3S4S5S6S7S8S9S0SJSQSKS';
VAR
  CardNum, CardToAdd : Integer;
  TempDeck : TypeDeck;
BEGIN
  Deck :=  '';
  TempDeck := FullDeck;
  Randomize;
  FOR CardNum := 1 to 52 DO
  BEGIN
    CardToAdd := Random(Length(TempDeck) DIV 2) * 2 + 1;
    Deck := Deck + Copy(TempDeck, CardToAdd, 2);
    TempDeck := Copy(TempDeck, 1, CardToAdd - 1) +
                Copy(TempDeck, CardToAdd + 2, +
                Length(TempDeck) - (CardToAdd + 2))
  END
END;

PROCEDURE DealCards(VAR ComputerFirstCard, Deck : TypeDeck;
                    VAR PlayerTotal, ComputerTotal,
                        PlayerNumAce, ComputerNumAce,
                        PlayerNumCard, ComputerNumCard : Integer);
BEGIN
  DealACard(ComputerFirstCard, Deck, PlayerTotal,
```

```
              PlayerNumCard, PlayerNumAce, False);
  DealACard(ComputerFirstCard, Deck, PlayerTotal,
              PlayerNumCard, PlayerNumAce, False);
  DealAcard(ComputerFirstCard, Deck, ComputerTotal,
              ComputerNumCard, ComputerNumAce, True);
  DealACard(ComputerFirstCard, Deck, ComputerTotal,
              ComputerNumCard, ComputerNumAce, True)
END;


  PROCEDURE SetupScreen;
  BEGIN
    ClrScr;
    GotoXY(4,  1);  Write('COMPUTER''s Hand');
    GotoXY(7, 12);  Write('YOUR Hand')
  END;

BEGIN  {SetupHand}
  PlayerTotal     := 0;
  ComputerTotal    := 0;
  PlayerNumAce    := 0;
  ComputerNumAce   := 0;
  PlayerNumCard   := 0;
  ComputerNumCard  := 0;
  SetupScreen;
  SetupDeck(Deck);
  DealCards(ComputerFirstCard, Deck, PlayerTotal, ComputerTotal,
  PlayerNumAce, ComputerNumAce, PlayerNumCard, ComputerNumCard)
END;
```

```
PROCEDURE DoPlayersHand(    ComputerFirstCard : TypeDeck;
                        VAR Deck : TypeDeck;
                        VAR PlayerTotal, PlayerNumAce,
                            PlayerNumCard : Integer);

VAR
  Stand  : Boolean;
  Choice : Char;
BEGIN {DoPlayersHand}
  Stand := False;
  WHILE NOT (PlayerTotal >= 21) AND NOT Stand DO
  BEGIN
    GotoXY(1, 23); ClrEol;
    Write('Press H for HIT or S for STAND : ');
    Choice := ReadKey;
    CASE Choice OF
      'H', 'h' : DealACard(ComputerFirstCard, Deck, PlayerTotal,
                           PlayerNumCard, PlayerNumAce, False);
      'S', 's' : Stand := True
    END
  END;
    IF (PlayerTotal > 21) THEN
    BEGIN
      TextColor(White + 128);
      GotoXY(17,12); Write('BUST !!!');
      TextColor(Green)
    END;
    IF (PlayerTotal = 21) AND (PlayerNumCard = 2) THEN
    BEGIN
      TextColor(White + 128);
      GotoXY(17,12); Write('BLACKJACK !!! ');
```

```
      TextColor(Green)
    END
  END;
PROCEDURE DoComputerHand(   ComputerFirstCard : TypeDeck;
                            VAR Deck : TypeDeck;
                            VAR ComputerTotal, ComputerNumAce,
                               ComputerNumCard : Integer);
BEGIN
  WHILE NOT (ComputerTotal >= 16) DO
  BEGIN
    DealACard(ComputerFirstCard, Deck, ComputerTotal,
              ComputerNumCard, ComputerNumAce, True);
    Delay(900)  {computer thinking time}
  END;
  IF (ComputerTotal > 21) THEN
  BEGIN
    TextColor(White + 128);
    GotoXY(20,1); Write('BUST !!!');
    TextColor(Green)
  END;
  IF (ComputerTotal = 21) AND (ComputerNumCard = 2) THEN
  BEGIN
    TextColor(White + 128);
    GotoXY(20,1); Write('BLACKJACK !!!');
    TextColor(Green)
  END
END;

PROCEDURE UpdateChips(VAR NumChips, BetAmount : Integer;
                          PlayerTotal, ComputerTotal : Integer);
```

```
BEGIN
  IF ((PlayerTotal > ComputerTotal) AND (PlayerTotal <= 21)) OR
     (ComputerTotal > 21) THEN
  BEGIN
    GotoXY(1,23); ClrEol;
    Sound(400); Delay(90);Sound(300); Delay(90); NoSound;
    GotoXY(1,23); Write('You WON ', BetAmount,' Chips !! ');
    NumChips := NumChips + BetAmount
  END
  ELSE
  BEGIN
    GotoXY(1,23); ClrEol;
    Sound(3000); Delay(80); NoSound;
    GotoXY(1,23); Write('You LOST ', BetAmount,' Chips !! ');
    NumChips := NumChips - BetAmount
  END;
  Write('Press ENTER '); ReadLn;
END;

PROCEDURE StartBlackJack;
BEGIN {start blackjack}
  GameOver := False;
  GetNumChips(NumChips);
  PlaceBet(BetAmount, NumChips, GameOver);
  WHILE (NOT GameOver) AND (NumChips > 0) DO
  BEGIN
    SetupHand(ComputerFirstCard, Deck, PlayerTotal, ComputerTotal,
              PlayerNumAce, ComputerNumAce, PlayerNumCard,
                                    ComputerNumCard);
    IF NOT (ComputerTotal = 21) THEN
```

```
    Doplayershand(ComputerFirstCard, Deck, PlayerTotal,
                      PlayerNumAce, PlayerNumCard);
    DrawCard(ComputerFirstCard, 1, True, True);
    IF NOT (PlayerTotal > 21) AND NOT ((PlayerTotal = 21)
                                AND (PlayerNumCard = 2))
    THEN DoComputerHand(ComputerFirstCard, Deck, ComputerTotal,
                      ComputerNumAce, ComputerNumCard);
    UpdateChips(NumChips, BetAmount, PlayerTotal, ComputerTotal);
    IF NumChips > 0
    THEN PlaceBet(BetAmount, NumChips, GameOver)
     ELSE
     BEGIN
       GotoXY(1, 23);
       Sound(2000); Delay(100); Sound(500); Delay(100); NoSound;
       WriteLn('YOU ARE BANKRUPT !!!  *>- GAME OVER -<*')
     END
  END;
END;

BEGIN
  REPEAT
    StartBlackJack;
    Write('Play Again Y/N '); Ch := ReadKey;
  UNTIL Ch IN ['N', 'n']
END.
```

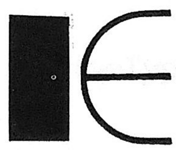
11.7.4 เกมยิงปีศาจอวกาศ

เกมนี้ดัดแปลงมาจากเกม BlindTargetShoot ในหนังสือ Hands-On Turbo Pascal เขียนโดย Larry Joel Goldstein

1. โหมดของจอภาพ เท็กซ์

2. กติกา มียานบินลึกลับลอบเข้ามาในเขตสุริยจักรวาล ผู้พิทักษ์โลกได้พยายามติดต่อสื่อสารทุกวิถีทางก็ไม่ได้รับคำตอบจากยานดังกล่าว ประกอบกับเครื่องวิเคราะห์ระยะไกลได้ข้อมูลเป็นที่แน่ชัดว่า ในยานนั้นมีอาวุธเต็มอัตรา จึงมีมติที่จะหยุดยานนั้นให้ได้ ปัญหามีอยู่ว่ายานนั้นจะปรากฏให้ตรวจจับได้ครั้งละ 2 วินาที แล้วยานก็จะหายไป ขุนศึกอวกาศได้รับคำสั่งให้ออกปฏิบัติการ โดยใช้อาวุธพิเศษที่มีช่วงการใช้งานเพียง 6 วินาที ถ้าไม่พบเป้าหมายภายในเวลาดังกล่าว อาวุธจะสลายไปทันที เพื่อป้องกันการทำลายผิดที่หมายและป้องกันการเกิดมลภาวะ ถ้าหาเป้าหมายเจออาวุธจะเข้าทำลายทันที และขณะที่ขุนศึกอวกาศกำลังจะออกปฏิบัติการ ข่าวล่าสุดก็แจ้งมาว่ายานลึกลับไม่ได้มีเพียงลำเดียว!

3. วิธีเล่น คอมพิวเตอร์จะแสดงยานลึกลับให้เห็นเป็นเวลาประมาณ 2 วินาที ลักษณะยานแสดงดังรูปที่ 11.9



รูปที่ 11.9 ภาพยานลึกลับที่หายตัวได้เหมือนปีศาจ

เมื่อภาพยานหายไปจะปรากฏเคอร์เซอร์ ซึ่งสมมติให้เป็นอาวุธอยู่บริเวณกลางจอภาพ เราจะต้องเลื่อนเคอร์เซอร์ไปยังตำแหน่งที่ยานอยู่ โดยใช้คีย์ลูกศรทั้ง 4 คีย์ ถ้าตรงเป้าหมายจะได้ยินเสียงยานถูกทำลาย แต่ถ้าไม่ตรงเป้าหมายหรือหมดเวลา จะมีรายงานให้ทราบว่าก่อนจะถูกทำลายอาวุธอยู่ห่างจากยานเป็นระยะทางเท่าไร

4. ข้อสังเกต ผู้เล่นเกมอาจจะใช้นิ้วชี้เป้าหมายบนจอภาพเอาไว้ก็ได้ แต่ถ้าตำแหน่งนั้นอยู่ห่างจากบริเวณกลางจอภาพมากเกินไปเวลาจะหมดก่อนที่อาวุธจะถึงเป้า

5. โปรแกรม รายละเอียดของโปรแกรมยิงปีศาจอวกาศ แสดงในโปรแกรม 11.4

โปรแกรม 11.4

```
PROGRAM ShootInvisibleUFO;
USES Crt,Dos;
VAR
   UfoX, UfoY, MyX, MyY : Integer;

PROCEDURE SetUpUFO(VAR UfoX,UfoY : Integer);
CONST
   UfoDisplayTime = 3;    {time in second}
BEGIN
   ClrScr;
   Randomize;
   UfoX := Random(50);
   UfoY := Random(25);
   GotoXY(UfoX, UfoY);
   TextColor(White + 128);
   Write(#178,#238);
   Delay(UfoDisplayTime * 1000);
   TextColor(Green);
   GotoXY(UfoX, UfoY);   Write(#32#32);
   Sound(800); Delay(10); NoSound;
END;

PROCEDURE Shoot (VAR MyX, MyY : Integer);
CONST
   TimeToMove   = 6; {time in second}
   ExtendedKey = #0;
   UpArrow      = #72;
   DownArrow    = #80;
```

```
  LeftArrow    = #75;
  RightArrow   = #77;
TYPE
  TimeType = RECORD
              Hour, Minute, Second, Sec100 : Word
           END;
VAR
  Timer : TimeType;
  Key   : Char;
  ArrowKeyPressed : Boolean;

  PROCEDURE StartTimer (VAR Timer : TimeType) ;
  BEGIN
     WITH Timer DO GetTime (Hour, Minute, Second, Sec100)
  END;

  FUNCTION ElapsedTime (Timer : TimeType) : Real;
  VAR
    Current : TimeType;
  BEGIN
    StartTimer(Current);
    IF (Current.Hour < Timer.Hour)
    THEN Current.Hour := Current.HOur + 24;
    IF (Current.Minute) < Timer.Minute
    THEN Current.Minute := Current.Minute + 60;
    IF (Current.Second < Timer.Second)
    THEN Current.Second := Current.Second + 60;
    IF (Current.Sec100 < Timer.Sec100)
    THEN Current.Sec100 := Current.Sec100 + 100;
```

```
    ElapsedTime := ((Current.Hour    - Timer.Hour) * 60 * 60) +
                   ((Current.Minute - Timer.Minute) * 60) +
                    (Current.Second - Timer.Second) +
                   ((Current.Sec100 - Timer.Sec100) / 100)
  END;

  PROCEDURE ReadArrowKey(VAR Key : Char;
                         VAR ArrowKeyPressed : Boolean);
  BEGIN
    IF NOT KeyPressed
    THEN ArrowKeyPressed := false
    ELSE
    BEGIN
      Key := ReadKey;
      IF (Key <> ExtendedKey)
      THEN ArrowKeyPressed := false
      ELSE
        BEGIN
          Key := ReadKey;
          ArrowKeyPressed := (Key IN[UpArrow, DownArrow,
                                      LeftArrow, RightArrow])
        END
      END
    END;

BEGIN {Shoot}
  MyX := 40;  {set cursor about screen center}
  MyY := 12;
  StartTimer(Timer);
  REPEAT
```

```
  GotoXY (MyX, MyY);
  IF (MyX = UfoX) AND (MyY = UfoY) THEN
  REPEAT          {fix cursor at the target}
    Sound(1000); Delay(10); NoSound;
    Sound(300); Delay(10); NoSound;
    GotoXY (MyX, MyY);
  UNTIL ElapsedTime(Timer) >= TimeToMove;
  ReadArrowKey(Key, ArrowKeyPressed);
  IF ArrowKeyPressed THEN
  BEGIN
    Sound(800); Delay(10); NoSound;
    CASE Key OF
      UpArrow    : IF (MyY = 1)
                   THEN MyY := 25
                   ELSE MyY := MyY - 1;
      DownArrow  : IF (MyY = 25)
                   THEN MyY := 1
                   ELSE MyY := MyY + 1;
      LeftArrow  : IF (MyX = 1)
                   THEN MyX := 80
                   ELSE MyX := MyX - 1;
      RightArrow : IF (MyX = 80)
                   THEN MyX := 1
                   ELSE MyX := MyX + 1
    END;
  END;
  UNTIL  ElapsedTime(Timer) >= TimeToMove
END; {Shoot}
```

```
PROCEDURE DisplayResult(UfoX, UfoY, MyX, MyY  : Integer) ;

   FUNCTION Distance( X1, Y1, X2, Y2 : Integer) : Integer;
   BEGIN
      Distance := Round(Sqrt(Sqr(X2 - X1) + Sqr(Y2 - Y1)))
   END;

BEGIN
   ClrScr;
   Write('Your distance from the target is : ');
   WriteLn(Distance(UfoX, UfoY, MyX, MyY));
   Write('Your score is : ');
   CASE Distance(UfoX, UfoY, MyX, MyY) OF
      0      : WriteLn('100 points!!!');
      1..2   : WriteLn(' 90 points!');
      3..5   : WriteLn(' 80 points.');
      6..10  : WriteLn(' 70 points.');
      11..20 : WriteLn(' 60 points.');
      21..35 : WriteLn(' 50 poits.');
      36..56 : WriteLn(' 40 poits.')
   ELSE
             WriteLn( ' 0 point.')
   END
END;

FUNCTION StopGame : Boolean;
VAR
   Ch : Char;
BEGIN
   Write('Do you want to play again (Y/N) ? ');
```

```
    ReadLn(Ch);
    StopGame := Ch IN['N','n'];
END;

BEGIN {MAIN}
    ClrScr;
    WriteLn('              IVISIBLE TARGET SHOOT');
    WriteLn('              ======================');
    WriteLn('A Random UFO Target will Light Up For 2 Seconds.');
    WriteLn('You must then move the cursor to the target based.');
    WriteLn('You have only five seconds to hit the target.');
    WriteLn('Press any key to Start Game.   GOOD LUCK TO YOU.');
    REPEAT UNTIL KeyPressed;
    REPEAT
        SetUpUFO(UfoX, UfoY);
        Shoot(MyX, MyY);
        DisplayResult(UfoX, UfoY, MyX, MyY)
    UNTIL StopGame
END.
```

11.7.5 เกมติกแต๊กโต (tictactoe)

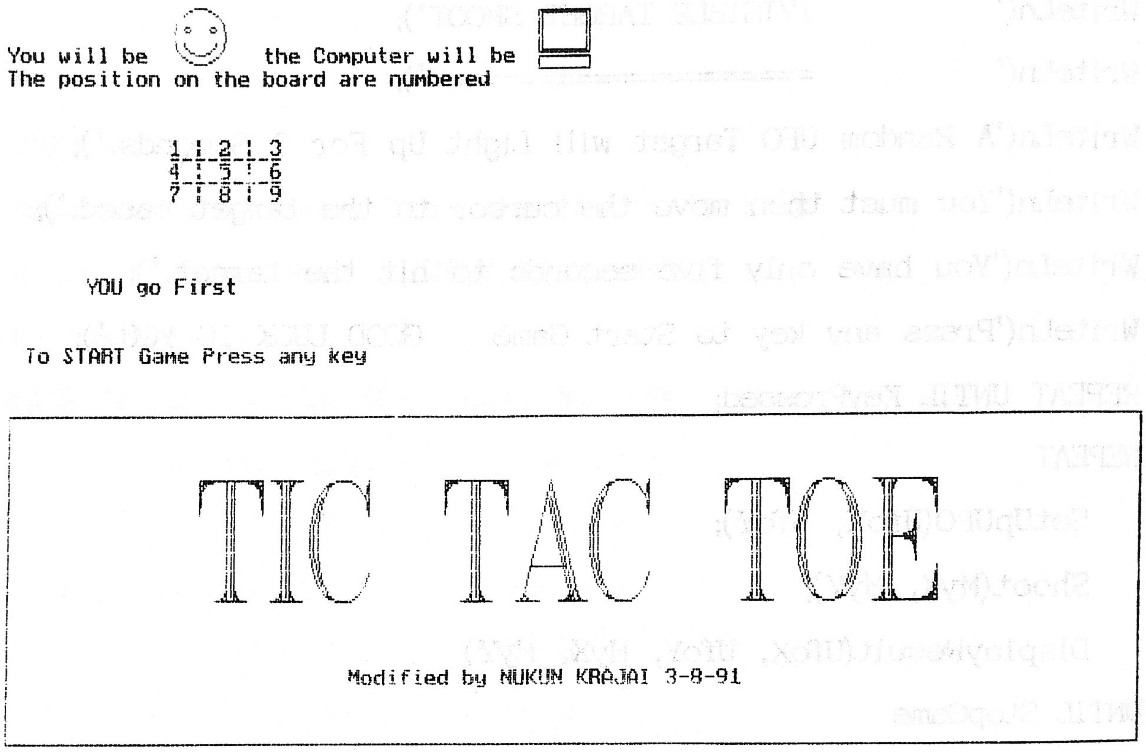
เกมนี้ดัดแปลงมาจากเกม tictactoe ในหนังสือ Hands-On Turbo Pascal เขียนโดย Larry Joel Goldstein

1. โหมดของจอภาพ กราฟิกส์

2. กติกา มีตารางสี่เหลี่ยมซึ่งมี 9 ช่อง ผู้เล่นมี 2 คน ให้ผู้เล่นแต่ละคนผลัดกันทำเครื่องหมายลงในตารางที่ยังว่าง ผู้ที่สามารถทำเครื่องหมายในตารางที่เรียงต่อกันในแนวเส้นตรงได้ก่อนจะเป็นผู้ชนะ เกมนี้เวลาเล่นบนกระดาษ มักจะกำหนดให้ผู้

เล่นคนหนึ่งใช้เครื่องหมาย X (เอ็กซ์) ส่วนอีกคนหนึ่งใช้เครื่องหมาย O (โอ) เกมนี้จึงมีอีกชื่อหนึ่งว่าเกมเอ็กซ์–โอ

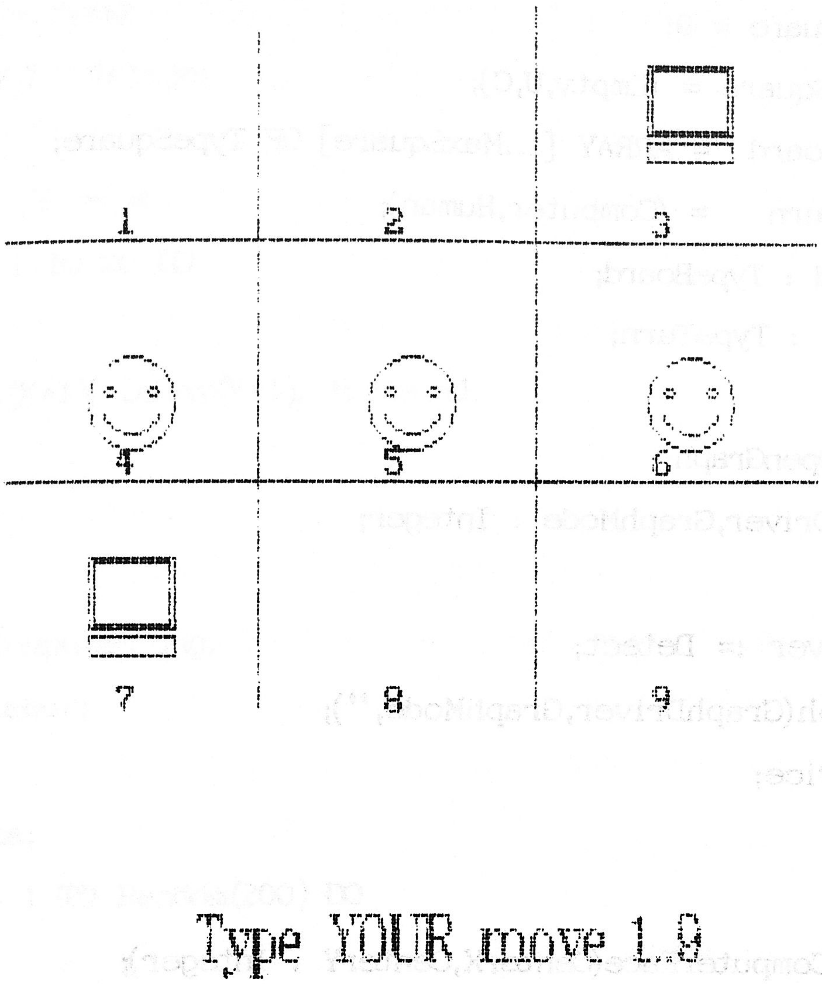
3. วิธีเล่น เกมนี้เป็นเกมประเภทคนเล่นกับคอมพิวเตอร์ เมื่อเริ่มเกมจะปรากฏที่จอภาพดังรูปที่ 11.10



รูปที่ 11.10 จอภาพเริ่มต้นของเกมซึ่งมีคำชี้แจงและบอกผู้เริ่มเล่น

เมื่อตกลงเล่นให้กดคีย์ ซึ่งที่จอภาพจะแสดงตารางให้เติมเครื่องหมายของผู้เล่นทั้งคู่ เมื่อเล่นไปสักครู่หนึ่งที่จอภาพอาจจะแสดงดังรูปที่ 11.11

ถ้าคอมพิวเตอร์เป็นผู้เริ่มก่อน เราจะเห็นภาพคอมพิวเตอร์ปรากฏอยู่ในตาราง 1 ตาราง และรอให้เราเลือกตารางบ้าง การเลือกทำเครื่องหมายในตารางทำโดยการกดคีย์ตัวเลขที่ต้องการ ซึ่งถ้าตารางนั้นยังว่างก็จะมีภาพหน้าคนปรากฏขึ้นในตาราง แต่ถ้าเลือกตารางไม่ว่าง จะไม่มีภาพเกิดขึ้นและต้องเลือกใหม่ ครั้งต่อไปคอมพิวเตอร์จะเป็นผู้เลือกบ้าง การเล่นจะสลับกันไปเช่นนี้จนมีผู้ชนะหรือเสมอกัน แล้วจึงเริ่มเกมใหม่ สำหรับผู้ชนะจะได้รับถ้วยรางวัลและบรรเลงเพลงประจำตัว



รูปที่ 11.11 ตารางเล่นเกมซึ่งคนเป็นผู้ชนะ

4. ข้อสังเกต เพื่อให้เกิดความแปลกในเกม จึงได้แทรกบทคอมพิวเตอร์โกงการเล่นไว้ด้วยคือ เมื่อเราเลือกตารางที่ 9 เอาไว้ก่อนแล้ว จะมีบางครั้งที่คอมพิวเตอร์มาเลือกซ้ำและกลายเป็นผู้ชนะ

5. โปรแกรม รายละเอียดของโปรแกรมเกมติกแต็กโตแสดงในโปรแกรม 11.5

โปรแกรม 11.5

```
PROGRAM TicTacToe;
USES  Crt,Graph;
```

```
CONST MaxSquare = 9;
Type  TypeSquare = (Empty,U,C);
      TypeBoard  = ARRAY [1..MaxSquare] OF TypeSquare;
      TypeTurn   = (Computer,Human);
VAR   Board : TypeBoard;
      Turn  : TypeTurn;

PROCEDURE OpenGraph;
VAR  GraphDriver,GraphMode : Integer;
BEGIN
   GraphDriver := Detect;
   InitGraph(GraphDriver,GraphMode,'');
   ClearDevice;
END;

PROCEDURE ComputerFace(CenterX,CenterY : Integer);
BEGIN
   Rectangle(CenterX-16,CenterY-16,CenterX+16,CenterY+2);
   Rectangle(CenterX-14,CenterY-15,CenterX+14,CenterY+1);
   Rectangle(CenterX-16,CenterY+4,CenterX+16,CenterY+9);
END;

PROCEDURE YourFace(CenterX,CenterY : Integer);
BEGIN
   Circle(CenterX,CenterY, 16);
   Arc(CenterX,CenterY,200,350,11);
   Circle(CenterX-8,CenterY-3,2);
   Circle(CenterX+8,CenterY-3,2);
END;
```

```
PROCEDURE YourSong;
VAR       X,Y,I : Integer;
BEGIN
  X := 80;  Y := 80;
  FOR I := 1 TO 20 DO
  BEGIN
    Sound(X*I); Delay(Y+I); Nosound;
  END;
END;

PROCEDURE ComputerSong;
VAR I : Integer;
BEGIN
  Randomize;
  FOR I := 1 TO Random(200) DO
  BEGIN
    Sound(Random(1200)); Delay(Random(15));Nosound;
  END;
END;

PROCEDURE InitGame(VAR Board : TypeBoard; VAR Turn : TypeTurn);
VAR  Dummy : Char;

  PROCEDURE InitBoard(VAR Board : TypeBoard);
  VAR  CurrentSquare : Integer;
  BEGIN
    FOR CurrentSquare := 1 TO MaxSquare DO
        Board[CurrentSquare] := Empty;
  END;
```

```
PROCEDURE FlipForTurn(VAR Turn : TypeTurn);
BEGIN
   Randomize;
   IF Random(2) = 1 THEN
   BEGIN
      OutTextXY(48,120,'YOU go First');
      Turn := Human
   END
   ELSE
   BEGIN
      OutTextXY(48,120,'COMPUTER go First');
      Turn := Computer
   END;
END;

PROCEDURE DrawBoard;
BEGIN
   ClearDevice;
   Line(103,8,103,191);
   Line(206,8,206,191);
   Line(8,70,311,70);
   OutTextXY(50,62,'1');
   OutTextXY(150,62,'2');
   OutTextXY(250,62,'3');
   OutTextXY(50, 124,'4');
   OutTextXY(150,124,'5');
   OutTextXY(250,124,'6');
   OutTextXY(50, 185,'7');
   OutTextXY(150,185,'8');
   OutTextXY(250,185,'9');
```

```
    Line(8,132,311,132);
  END;

BEGIN {initgame}
  ClearDevice;
  SetTextStyle(TriplexFont,HorizDir,8);
  OutTextXY( 120, 190,'TIC TAC TOE');
  Rectangle(0,180,GetMaxX,GetMaxY-25);
  SetTextStyle(DefaultFont,HorizDir,1);
  OutTextXY( 215, 290,'Modified by NUKUN KRAJAI 3-8-91');
  OutTextXY(  0, 20,'You will be');            YourFace(120,15);
  OutTextXY(160, 20,'the Computer will be'); ComputerFace(350,20);
  OutTextXY(  0, 30,'The position on the board are numbered');
  OutTextXY( 100, 60,'1 ¦ 2 ¦ 3');
  OutTextXY( 100, 65,'---------');
  OutTextXY( 100, 70,'4 ¦ 5 ¦ 6');
  OutTextXY( 100, 75,'---------');
  OutTextXY( 100, 80,'7 ¦ 8 ¦ 9');
  FlipForTurn(Turn);
  OutTextXY( 10,150,'To START Game Press any key');
  Dummy := ReadKey;
  InitBoard(Board);
  DrawBoard;
END;

PROCEDURE ClearMsg;
BEGIN
  SetFillStyle(EmptyFill,1);
  Bar(0,240,319,330);
END;
```

```
FUNCTION Won(Board : TypeBoard; Side : TypeSquare) : Boolean;
BEGIN
  Won :=
  ((Board[1]=Board[2]) AND (Board[2]=Board[3]) AND (Board[3]=Side)) OR
  ((Board[4]=Board[5]) AND (Board[5]=Board[6]) AND (Board[6]=Side)) OR
  ((Board[7]=Board[8]) AND (Board[8]=Board[9]) AND (Board[9]=Side)) OR
  ((Board[1]=Board[4]) AND (Board[4]=Board[7]) AND (Board[7]=Side)) OR
  ((Board[2]=Board[5]) AND (Board[5]=Board[8]) AND (Board[8]=Side)) OR
  ((Board[3]=Board[6]) AND (Board[6]=Board[9]) AND (Board[9]=Side)) OR
  ((Board[1]=Board[5]) AND (Board[5]=Board[9]) AND (Board[9]=Side)) OR
  ((Board[3]=Board[5]) AND (Board[5]=Board[7]) AND (Board[7]=Side))
END;

PROCEDURE PlayGame(VAR Board : TypeBoard; Turn : TypeTurn);

   PROCEDURE FindCenter( Square : Integer;
                         VAR CenterX,CenterY : Integer);
   BEGIN
     CASE Square OF
        1,2,3 : CenterY := 5 * 8;
        4,5,6 : CenterY :=14 * 8;
        7,8,9 : CenterY :=21 * 8;
     END;
     CASE Square OF
        1,4,7 : CenterX := 7 * 8;
        2,5,8 : CenterX :=20 * 8;
        3,6,9 : CenterX :=33 * 8;
     END;
  END;
```

```
PROCEDURE DrawComp(Square : Integer);
VAR CenterX,CenterY : Integer;
BEGIN
   FindCenter(Square,CenterX,CenterY);
   Rectangle(CenterX-16,CenterY-16,CenterX+16,CenterY+2);
   Rectangle(CenterX-14,CenterY-15,CenterX+14,CenterY+1);
   Rectangle(CenterX-16,CenterY+4,CenterX+16,CenterY+9);
END;

PROCEDURE DrawFace(Square : Integer);
VAR CenterX,CenterY : Integer;
BEGIN
   FindCenter(Square,CenterX,CenterY);
   Circle(CenterX,CenterY, 16);
   Arc(CenterX,CenterY,200,350,11);
   Circle(CenterX-8,CenterY-3,2);
   Circle(CenterX+8,CenterY-3,2);
END;

PROCEDURE DoComputerMove(VAR Board : TypeBoard; VAR Turn : TypeTurn);
VAR Move : Integer;

  PROCEDURE FindMove(Board : TypeBoard; VAR Move : Integer);

    FUNCTION ComputerWillWin(Board : TypeBoard;Move : Integer):
                                 Boolean;
    BEGIN
      Board[Move] := C;
      ComputerWillWin := Won(Board,C);
    END;
```

```
    PROCEDURE FindWinningMove(Board : TypeBoard; VAR Move : Integer);
    BEGIN
       REPEAT
          REPEAT
             Inc(Move);
          UNTIL (Board[Move] = Empty) OR (Move = MaxSquare)
       UNTIL (Move = MaxSquare) OR ComputerWillWin(Board,Move)
    END;

    PROCEDURE FindRandomMove(Board : TypeBoard; VAR Move : Integer);
    BEGIN
       REPEAT
          Move := Random(MaxSquare) + 1;
       UNTIL Board[Move] = Empty
    END;

  BEGIN  {findmove}
     Move := 1;
     FindWinningMove(Board,Move);
     IF NOT ComputerWillWin(Board,Move) THEN FindRandomMove(Board,Move);
  END;

BEGIN {docomputermove}
   FindMove(Board,Move);
   ClearMsg;
   OutTextXY(80,240,'Here is MY move !');
   Delay(600);
   Board[Move] := C;
   DrawComp(Move);   Sound(500);Delay(50);NoSound;
   Turn := Human;
END;
```

```
PROCEDURE DoYouMove(VAR Board : TypeBoard; VAR Turn : TypeTurn);
VAR Move : Integer;
BEGIN
  ClearMsg;
  OutTextXY(80,240,'Type YOUR move 1..9 ');
  REPEAT
    REPEAT
      Move := Ord(ReadKey) - Ord('0')
    UNTIL Move IN [1..9]
  UNTIL Board[Move] = Empty;
  Board[Move] := U;
  DrawFace(Move);    Sound(1000);Delay(50);NoSound;
  Turn := Computer;
END;

FUNCTION Full( Board : TypeBoard) : Boolean;
VAR CurrentSquare : Integer;
BEGIN
  Full := True;
  FOR CurrentSquare := 1 TO MaxSquare DO
  IF Board[CurrentSquare] = Empty THEN Full := False
END;

BEGIN  {playgame}
  REPEAT
    IF Turn = Computer THEN DoComputerMove(Board,Turn)
                       ELSE DoYouMove(Board,Turn)
  UNTIL Won(Board,U) OR Won(Board,C) OR Full(Board)
END;
```

```
PROCEDURE Award;
VAR   CenterX,CenterY,SizeX,SizeY   : Integer;
BEGIN
  SizeX   := 40;      SizeY  := 50;
  CenterX := 500;  CenterY  := 50;
  Ellipse(CenterX,CenterY,180,360,SizeX,SizeY);
  Line(CenterX-SizeX,CenterY,CenterX+SizeX,CenterY);
  Rectangle(CenterX-SizeX+10,CenterY+100,CenterX+SizeX-10,
            CenterY+105);
  Rectangle(CenterX-SizeX+5,CenterY+105,CenterX+SizeX-5,
            CenterY+120);
  Rectangle(CenterX-SizeX+35,CenterY+SizeY,CenterX+SizeX-35,
            CenterY+100);
  Ellipse(CenterX-SizeX-10,CenterY+5,0,360,SizeX-30,SizeY-30);
  Ellipse(CenterX+SizeX+10,CenterY+5,0,360,SizeX-30,SizeY-30);
  Ellipse(CenterX-11,CenterY-110,0,360,20,35);
  Ellipse(CenterX+30,CenterY-110,0,360,20,30);
  Ellipse(CenterX+65,CenterY-90,0,360,17,28);
  Ellipse(CenterX+90,CenterY-80,0,360,10,25);
END;

PROCEDURE DisplayOutcome(Board : TypeBoard);
BEGIN
  ClearMsg;
  IF Won(Board,U) THEN
  BEGIN
    OutTextXY(0,240,'YOU WIN ! ');
    Award;YourFace(500,73); YourSong
  END
  ELSE
```

```
IF Won(Board,C)THEN
BEGIN
  OutTextXY(0,240,'I WIN ! ');
  Award;ComputerFace(500,75); ComputerSong
END
  ELSE OutTextXY(0,240,'WHO WIN ? !')
END;

FUNCTION Done : Boolean;
VAR  Answer : Char;
BEGIN
  OutTextXY(100,240,'Play Again Y/N ');
  Answer := ReadKey;
  Done := Answer IN ['N','n']
END;

BEGIN  {MAIN}
  OpenGraph;
  REPEAT
     InitGame(Board,Turn);        SetTextStyle(TriplexFont,HorizDir,1);
     PlayGame(Board,Turn);
     DisplayOutCome(Board);
  UNTIL Done;
  CloseGraph
END.
```

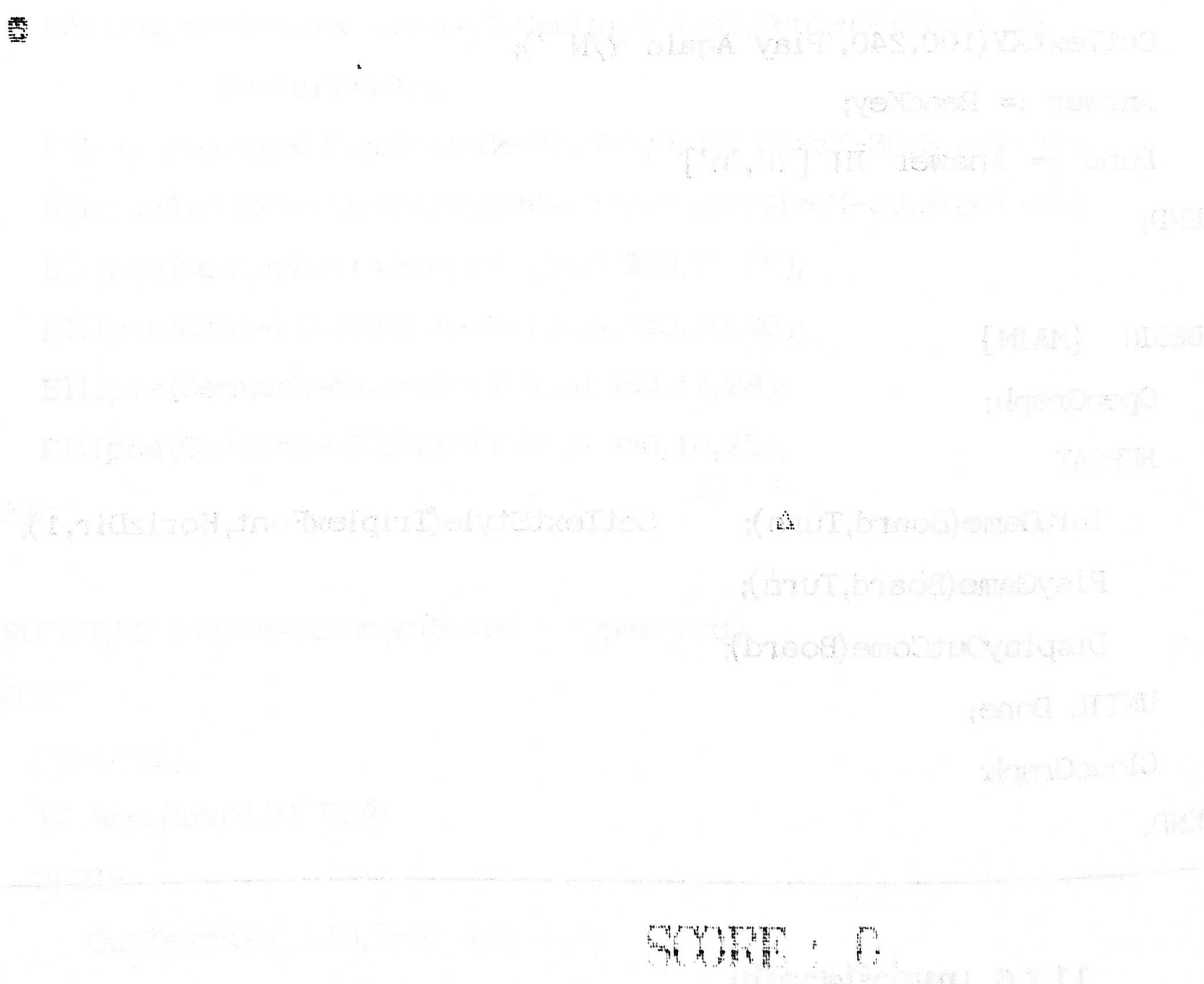
11.7.6 เกมล่าปีศาจบิน

เกมนี้ดัดแปลงมาจากเกม shooting gallery ในหนังสือ Hands-On Turbo Pascal เขียนโดย Larry Joel Goldstein

1. โหมดของจอภาพ กราฟิกส์

2. กติกา เกมนี้เหมือนกับเกมยิงเป้าในสวนสนุกคือ มีเป้าเป็นรูปหน้าคนออกมาจากทางด้านซ้ายของจอภาพ แล้วเคลื่อนไปทางขวาของจอภาพและหายไปที่ขอบขวาของจอภาพ เราจะต้องใช้ปืนยิงเป้านี้ให้ถูก ถ้ายิงไม่ถูกเป้าจะเลยไปจากจอภาพและวนกลับมาใหม่ ถ้ายิงถูกเป้าจะหายไปทันทีและจะมีเป้าใหม่ออกมาแทน เป้าใหม่ทุกเป้าจะเพิ่มความเร็วขึ้นเรื่อยๆ ผู้ที่ยิงถูกเป้ามากครั้งจะได้คะแนนมากกว่า

3. วิธีเล่น เมื่อเกมเริ่ม เราจะเห็นเป้าเคลื่อนที่วนจากขอบจอภาพทางซ้ายไปยังขอบจอภาพทางขวา บริเวณกลางจอภาพมีปืนวางอยู่ ดังแสดงในรูปที่ 11.12



รูปที่ 11.12 จอภาพของเกมล่าปีศาจบิน

ปืนสามารถเลื่อนได้ในแนวซ้ายขวาโดยใช้คีย์ลูกศร ตอนยิงให้กดคีย์วรรคเราจะต้องคอยสังเกตเป้า กะระยะให้ดี แล้วยิง ถ้ายิงถูกคะแนนจะเพิ่มขึ้น เป้าใหม่ที่เร็วกว่าจะเคลื่อนมาอีก เราก็ยิงอีก พยายาม ยิง ยิง และยิงให้ถูกเป้ามากที่สุด เพื่อจะได้เป็นผู้ชนะ

4. ข้อสังเกต เนื่องจากปืนอยู่ห่างจากเป้ามาก เราจึงต้องยิงล่วงหน้าก่อนที่เป้าจะเข้ามาอยู่ในแนวกระสุน มิฉะนั้นจะยิงไม่ถูก ความเร็วที่เพิ่มขึ้นของเป้าใหม่จะทำให้เรากะระยะผิด ประกอบกับปืนก็เคลื่อนที่ได้อย่างล่าช้า ในขณะที่เป้าสามารถทวีความเร็วขึ้นได้เรื่อยๆ แล้วเราจะแก้ปัญหานี้ได้อย่างไร

5. โปรแกรม รายละเอียดของโปรแกรมล่าปีศาจบิน แสดงในโปรแกรม 11.6

โปรแกรม 11.6

```
PROGRAM ShootingSprite;
USES Crt,Graph;
CONST
  PictureSize    = 20;
  PictureHeight  = 8;
  PictureWidth   = 8;
  NumTargets     = 1000;  {maximum score}
TYPE
  TypePicture   = ARRAY [1.. PictureSize] OF Byte;
  TypeSprite    = RECORD
                    X,Y     : Integer;
                    Picture : TypePicture;
                  END;
```

```
VAR
  Gun, Bullet, Target  : TypeSprite;
  Score, TargetNumber, Speed  : Integer;
  BulletInMotion, TargetHit : Boolean;

PROCEDURE DisplaySprite(VAR Sprite : TypeSprite;
                        XStart,YStart : Integer);
BEGIN
  WITH Sprite DO
  BEGIN
    IF (XStart >= 0) and (XStart <= (getmaxx - PictureWidth))
    THEN X := XStart ELSE X := 0;
    IF (YStart >= 0) and (Ystart <= (getmaxy - PictureHeight))
    THEN Y := YStart ELSE Y := 0;
    PutImage(X, Y, Picture, XorPut)
  END
END;

PROCEDURE RemoveSprite(VAR Sprite : TypeSprite);
BEGIN
   WITH Sprite DO PutImage(X, Y, Picture, XorPut)
END;

PROCEDURE MoveSprite(VAR Sprite : TypeSprite;
                     XRel, YRel : Integer);
BEGIN
  RemoveSprite(Sprite);
  WITH Sprite DO
  BEGIN
    IF ((X + XRel) >= 0) AND
```

```
      ((X + XRel) <= (GetMaxX - PictureWidth))
    THEN X := X + XRel;
    IF ((Y + YRel) >= 0) AND
      ((Y + YRel) <= (GetMaxY - PictureHeight))
    THEN Y := Y + YRel;
    PutImage(X, Y, Picture, XorPut)
  END
END;

PROCEDURE DisplayScore( Score : Integer);
VAR
  ScoreString : STRING;
BEGIN
  Str(Score, ScoreString);
  SetTextStyle(TriplexFont,HorizDir,1);
  OutTextXY(250,300,'SCORE :');
  ScoreString := ScoreString;
  SetFillStyle(EmptyFill, 1);
  Bar(330, 300, 380, 345);
  OutTextXY(330,300,ScoreString)
END;

PROCEDURE InitGame(VAR Gun, Bullet, Target : TypeSprite;
                   VAR Score : Integer;
                   VAR BulletInMotion : Boolean);

  PROCEDURE OpenGraph;
  VAR
    GraphDriver, GraphMode : Integer;
```

```
  BEGIN
     GraphDriver := Detect;
     InitGraph(GraphDriver, GraphMode, '');
     ClearDevice
  END;

  PROCEDURE CreatePictures(VAR Gun, Bullet, Target : TypeSprite);
  BEGIN
     SetColor(1);
     OutText(#127);   {gun}
     GetImage(0, 0, (PictureWidth - 1), (PictureHeight - 1),
                     Gun.Picture);
     ClearDevice;
     SetColor(2);
     OutText(#124);    {bullet}
     GetImage(0, 0, (PictureWidth -1), (PictureHeight -1),
                     Bullet.Picture);
     ClearDevice;
     SetColor(3);
     OutText(#2);     {target}
     GetImage(0, 0, (PictureWidth - 1), (PictureHeight - 1),
                     Target.Pictur;
     ClearDevice
  END;

BEGIN
  OpenGraph;
  CreatePictures(Gun, Bullet, Target);
  DisplaySprite(Gun, 300, 169);
  Score := 0;
```

```
    DisplayScore(Score);
    BulletInMotion := False
END;

PROCEDURE PlayARound(VAR Gun, Bullet, Target : TypeSprite;
                    VAR BulletInMotion, TargetHit : Boolean);
VAR
  Column : Integer;

  PROCEDURE InitRound(VAR Target : TypeSprite;
                      VAR TargetHit : Boolean);
  BEGIN
    DisplaySprite(Target, 0, 8);
    TargetHit := False
  END;

  FUNCTION Even(Number : Integer) : Boolean;
  BEGIN
    Even := Not Odd(Number)
  END;

  PROCEDURE DoHit(VAR Target : TypeSprite;
                  VAR TargetHit : Boolean;
                  VAR Score : Integer);
  VAR Turn : Integer;
  BEGIN
    FOR Turn := 1 TO 4 DO    {burst sound}
    BEGIN
```

```
      Sound(3500); Delay(20); Sound(500); Delay(50);
   END;
   NoSound;
   Dec(Speed);    {speed up 1 step}
   RemoveSprite(Target);
   TargetHit := true;
   Score := Score + 1
END;

PROCEDURE CheckBullet(VAR Bullet, Target : TypeSprite;
                      VAR BulletInMotion, TargetHit : Boolean;
                      VAR Score : Integer);
BEGIN
   MoveSprite(Bullet, 0, -PictureHeight);
   IF (Bullet.Y <= (PictureHeight + 8)) THEN
   BEGIN
      RemoveSprite(Bullet);
      BulletInMotion := false;
      IF (abs(Bullet.X - Target.X) < 7) THEN
      DoHit(Target, TargetHit, Score)
   END
END;

PROCEDURE CheckKeyboard(VAR Gun, Bullet : TypeSprite;
                        VAR BulletInMotion : Boolean);
CONST
   ExtendedKey = #0;
   SpaceBar    = #32;
   LeftArrow   = #75;
   RightArrow  = #77;
```

```
  Esc          = #27;
VAR
  Key : Char;
BEGIN
  IF KeyPressed THEN
  BEGIN
    Key := Readkey;
    IF Key = Esc THEN BEGIN CloseGraph; Halt END;
    IF Key = SpaceBar THEN        {shoot}
    IF NOT BulletInMotion THEN
    BEGIN
        Sound(3500); Delay(20); NoSound;
        DisplaySprite(Bullet, Gun.X, 169);
        BulletInMotion := True
     END;
     IF Key = ExtendedKey THEN
     CASE ReadKey OF           {move gun}
        LeftArrow  : MoveSprite(Gun, -8, 0);
        RightArrow : MoveSprite(Gun, +8, 0)
     END
   END
 END;

BEGIN {PlayARound}
  InitRound(Target, TargetHit);
  FOR Column := 2 TO (GetMaxX - PictureWidth) DO
  IF Even (Column) AND NOT TargetHit THEN
  BEGIN
    MoveSprite(Target, +2, 0);
    IF BulletInMotion THEN
```

```
      CheckBullet(Bullet, Target, BulletInMotion, TargetHit, Score);
      CheckKeyboard(Gun, Bullet, BulletInMotion);
      Delay (Speed);  {speed of target}
   END;
END;

PROCEDURE DetermineResults(VAR Target : TypeSprite;
                               TargetHit : Boolean;
                           VAR Score : Integer);
BEGIN
   IF NOT TargetHit THEN RemoveSprite(Target);
   DisplayScore(Score)
END;

BEGIN {ShootingBalloon}
   Speed := 50;  {start speed, slowest}
   InitGame(Gun, Bullet, Target, Score, BulletInMotion);
   FOR TargetNumber := 1 TO NumTargets DO
   BEGIN
      PlayARound(Gun, Bullet, Target, BulletInMotion, TargetHit);
      DetermineResults(Target, TargetHit, Score)
   END;
   CloseGraph
END.
```

11.7.7 เกมทาวเวอร์ออฟฮานอย (Tower of Hanoi)

เกมนี้ดัดแปลงมาจากเกม Towers of Hanoi ในหนังสือ Hands-On Turbo Pascal เขียนโดย Larry Joel Goldstein

1. โหมดของจอภาพ เท็กซ์

2. กติกา ทาวเวอร์ออฟฮานอยประกอบด้วยหลัก 3 หลัก ที่หลักที่ 1 ซึ่งอยู่ทางซ้ายมือมีแป้นกลมซึ่งมีรูตรงกลาง (disk) สวมอยู่ แป้นทุกอันมีขนาดต่างกันวางซ้อนกันตามลำดับขนาด ให้แป้นอันใหญ่สุดอยู่ล่างสุดและแป้นอันเล็กสุดอยู่บนสุด ผู้เล่นเกมจะต้องย้ายแป้นจากหลักที่ 1 ให้ไปอยู่ที่หลักที่ 3 ซึ่งเป็นหลักอยู่ทางขวาสุด โดยให้เรียงลำดับเหมือนเดิม การย้ายแป้นมีเงื่อนไข 2 ประการคือ จะต้องย้ายครั้งละ 1 แป้น และห้ามวางแป้นอันใหญ่บนแป้นอันเล็ก ผู้ทำได้สำเร็จเป็นผู้ชนะ

3. วิธีเล่น เนื่องจากเกมนี้จะแก้ปัญหาได้ยากขึ้นตามจำนวนแป้นที่เพิ่มขึ้น ดังนั้น โปรแกรมจึงแสดงการแก้ปัญหาให้ดูเป็นตัวอย่างด้วย เมื่อเริ่มเกมที่จอภาพจะเป็นดังรูปที่ 11.13

```
==========================================

              TOWERS OF HANOI

------------------------------------------

W_atch The Computer Solve The Puzzle

S_olve The Puzzle Yourself

Q_uit


          Please select:  W  S  Q _
```

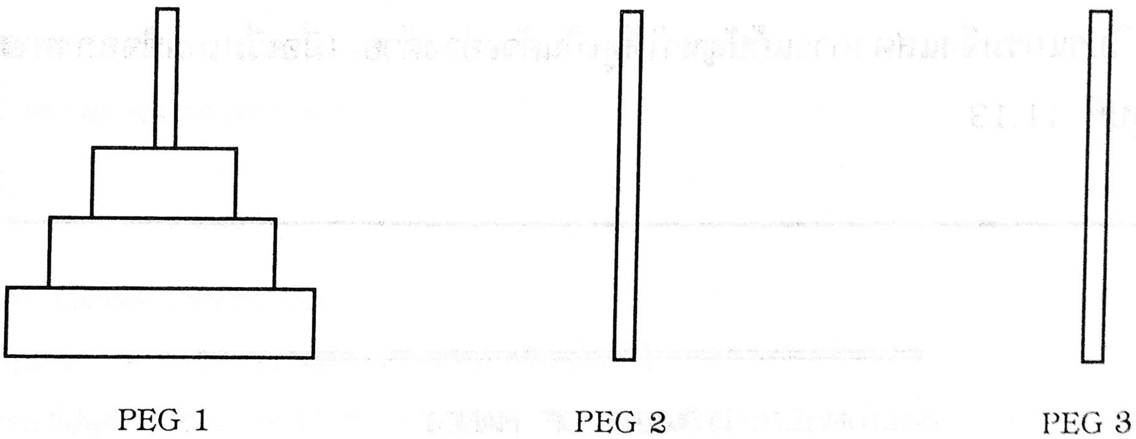
รูปที่ 11.13 จอภาพเริ่มต้นของเกมทาวเวอร์ออฟฮานอย

ถ้ากดคีย์ W แสดงว่าเราต้องการให้คอมพิวเตอร์แก้ปัญหาให้ดู ถ้ากดคีย์ S แสดงว่าเราจะแก้ปัญหาเอง เมื่อเลือก W หรือ S แล้ว คอมพิวเตอร์จะถามจำนวนแป้นที่เราต้องการ ดังต่อไปนี้

You may use 1 to 10 disks.

Please enter number of disks (1..10) _

จำนวนแป้นมีให้เลือกตั้งแต่ 1 ถึง 10 แป้น เมื่อเลือกแล้ว สมมติว่าเลือก 3 แป้น ที่จอภาพจะเป็นดังรูปที่ 11.14



รูปที่ 11.14 ทาวเวอร์ออฟฮานอยที่มี 3 แป้น

ถ้าเราเลือกให้คอมพิวเตอร์แสดงการแก้ปัญหา ที่ด้านล่างของจอภาพจะมีข้อความบอกว่ากำลังย้ายแป้นจากหลัก (PEG) ไหน ไปที่หลักไหน พร้อมกับแสดงภาพให้ดู โดยจะทำเช่นนี้ไปจนได้ผลตามต้องการแล้วจึงหยุด

ถ้าเราเลือกแก้ปัญหาเอง ที่ด้านล่างของจอภาพจะมีข้อความถามว่า ต้องการเลื่อนแป้นจากหลักที่เท่าไร ดังนี้

Move FROM PEG 1, 2, 3 or Q (Quit) : _

เมื่อเรากดคีย์ 1, 2 หรือ 3 แล้ว จะมีข้อความถามว่า จะเลื่อนไปหลักที่เท่าไร ดังนี้

Move TO PEG 1, 2, or 3 _

เมื่อเราเลือกหลักแล้ว ถ้าไม่ผิดกติกา แป้นจะถูกเลื่อนไปยังหลักนั้น แต่ถ้าผิดกติกาจะมีข้อความบอกว่าเลื่อนไม่ได้ ดังนี้

INVALID Move

และจะเริ่มถามว่าจะเลื่อนแป้นจากหลักที่เท่าไรซ้ำอีก เมื่อทำเช่นนี้ไปจนสำเร็จจะมีข้อความขอแสดงความยินดี ดังนี้

CONGRATULATION ! You Solved It !

4. ข้อสังเกต การแก้ปัญหานี้เป็นสิ่งที่ท้าทายความสามารถและเป็นเกมที่ต้องใช้ความตั้งใจสูงมาก โดยเฉพาะอย่างยิ่งเมื่อมีจำนวนแป้นมากขึ้น ดังนั้นจึงควรเริ่มตั้งแต่ 3 แป้นก่อน เมื่อแก้ปัญหาได้แล้วจึงค่อยเพิ่มจำนวนขึ้นไปเรื่อยๆ

5. โปรแกรม รายละเอียดของโปรแกรมเกมทาวเวอร์ออฟฮานอยแสดงในโปรแกรม 11.7

โปรแกรม 11.7

```
PROGRAM TowerOfHanoi;
USES
  Crt;
CONST
  MaxItems   = 10;
  Quit        = 'q';
  BaseLevel = 16;
  Disk        = #220;
TYPE
  TypeItem    = Integer;
  TypeStackArray = ARRAY[1..MaxItems] OF TypeItem;
  TypeStack = RECORD
```

```
                StackArray : TypeStackArray;
                TopOfStack : Integer
              END;
   TypePeg  =  RECORD
                Stack      : TypeStack;
                PegNumber : Integer
              END;
VAR
   Choice : Char;

PROCEDURE Center(S : STRING);
VAR Col : Integer;
BEGIN
   FOR Col := 1 TO (80 - Length(S)) DIV 2 DO
   Write(' ');
   Write(S)
END;

FUNCTION EmptyStack(Stack : TypeStack) : Boolean;
BEGIN
   EmptyStack := (Stack.TopOfStack = 0)
END;

FUNCTION FullStack(Stack : TypeStack) : Boolean;
BEGIN
   FullStack := (Stack.TopOfStack = MaxItems)
END;

PROCEDURE InitializeStack(VAR Stack : TypeStack);
BEGIN
```

```
  Stack.TopOfStack := 0
END;

PROCEDURE Push(VAR Stack : TypeStack;
               Item : Integer);
BEGIN
  IF NOT FullStack(Stack) THEN
  WITH Stack DO
  BEGIN
    TopOfStack := (TopOfStack + 1);
    StackArray[TopOfStack] := Item
  END
END;

PROCEDURE Pop (VAR Stack : TypeStack;
          VAR Item : Integer);
BEGIN
  IF not EmptyStack(Stack) THEN
  WITH Stack DO
  BEGIN
    Item := StackArray[TopOfStack];
    TopOfStack := (TopOfStack - 1)
  END
END;

PROCEDURE ShowMenuAndGetChoice(VAR Choice : Char);
BEGIN
  ClrScr;  GotoXY(1,6);
  Center('====================================');
  WriteLn;
```

```
   Center('  TOWERS OF HANOI');
   WriteLn;
   Center('-------------------------------------------');
   WriteLn;
   Center('W_atch The Computer Solve The Puzzle');
   WriteLn;
   Center('S_olve The Puzzle Yourself          ');
   WriteLn;
   Center('Q_uit                               ');
   WriteLn;
   WriteLn;
   REPEAT
      GotoXY(1,13); ClrEol;
      Center('Please select  W  S  Q ');
      Choice := ReadKey;
   UNTIL Choice IN['W','w','S','s','Q','q']
END;

FUNCTION NumberInStack(Stack : TypeStack) : Integer;
VAR
   Item : TypeItem;
BEGIN
   IF not EmptyStack(Stack) THEN
   BEGIN
      Pop(Stack, Item);
      NumberInStack := (NumberInStack(Stack) + 1)
   END
   ELSE
      NumberInStack := 0
END;
```

```
PROCEDURE InitializePeg(VAR Peg : TypePeg;
                        Number : Integer);

BEGIN
  WITH Peg DO
  BEGIN
    PegNumber := Number;
    InitializeStack(Stack)
  END
END;

PROCEDURE InitializePegs (VAR StartPeg, IntermediatePeg,
                          EndPeg : Typepeg);
BEGIN
  InitializePeg(StartPeg, 1);
  InitializePeg(IntermediatePeg, 2);
  InitializePeg(EndPeg, 3)
END;

PROCEDURE GetNumberOfDisks(VAR NumberOfDisks : Integer);
BEGIN
  REPEAT
    GotoXY(1,15);
    Center('You may use 1 to 10 disks. ');
    WriteLn;  ClrEol;
    Center('Please enter number of disks (1..10) ');
    ReadLn(NumberOfDisks)
  UNTIL (NumberOfDisks in [1..10])
END;
```

```
PROCEDURE FillStartPeg(VAR StartPeg : TypePeg;
                           DiskNumber : Integer);
BEGIN
  IF (DiskNumber > 0) THEN
  BEGIN
    Push(StartPeg.Stack, DiskNumber);
    FillStartPeg(StartPeg, DiskNumber - 1)
  END
END;

PROCEDURE WriteRepeatedChars( CharacterToWrite : Char;
                              NumberToWrite : Integer);
VAR
  Counter : Integer;
BEGIN
  FOR Counter := 1 to NumberToWrite DO
  Write(CharacterToWrite)
END;

PROCEDURE WriteBase( Column : Integer);
BEGIN
  GotoXY(Column - 10, BaseLevel);
  WriteRepeatedChars(' ',21)
END;

PROCEDURE LabelPeg(  PegNumber,Column : Integer);
BEGIN
  GotoXY(Column - 2, BaseLevel + 2);
  WriteLn('PEG ', PegNumber)
END;
```

```
PROCEDURE WriteStick(  Column : Integer);
VAR
  Counter : Integer;
BEGIN
  FOR Counter := 1 to (BaseLevel - 1) DO
  BEGIN
    GotoXY(Column, Counter);
    Write(' ')
  END
END;

PROCEDURE WriteDisks( Peg : TypePeg;
                      Column : Integer);
VAR
  Row, DiskNumber : Integer;
  Item : TypeItem;
BEGIN
  WITH Peg DO
  BEGIN
    Row := (BaseLevel - 1);
    FOR DiskNumber := NumberInStack(Stack) downto 1 DO
    BEGIN
      TextColor(LightCyan);
      GotoXY(Column - DiskNumber, Row);
      WriteRepeatedChars(Disk, 1 + (DiskNumber * 2));
      Row := Row - 1;
      TextColor(LightGray)
    END
  END
END;
```

```
PROCEDURE WritePeg(Peg : TypePeg);
VAR
  Column, Row, Bottom : Integer;
BEGIN
  WITH Peg DO
  BEGIN
    Column := (11 + ((PegNumber - 1) * 26));
    WriteBase(Column);
    LabelPeg(PegNumber, Column);
    WriteStick(Column);
    WriteDisks(Peg, Column)
  END
END;

PROCEDURE WritePegs(StartPeg, IntermediatePeg, EndPeg : TypePeg);
BEGIN
  WritePeg(StartPeg);
  WritePeg(IntermediatePeg);
  WritePeg(EndPeg);
END;

PROCEDURE SetUp (VAR StartPeg, IntermediatePeg, EndPeg : TypePeg);
VAR
  NumberOfDisks : Integer;
BEGIN
  InitializePegs(StartPeg, IntermediatePeg, EndPeg);
  GetNumberOfDisks(NumberOfDisks);
  FillStartPeg(StartPeg, NumberOfDisks);
  ClrScr;
```

```
  WritePegs(StartPeg, IntermediatePeg, EndPeg)
END;

PROCEDURE SayMoveFrom(FromDisk, ToDisk : Integer);
VAR FDiskStr,TDiskStr : STRING[1];
BEGIN
  Str(FromDisk,FDiskStr);
  Str(ToDisk,TDiskStr);
  GotoXY(1, 21); ClrEol;
  Center('Move Disk From '+ FDiskStr + ' to ' + TDiskStr)
END;

PROCEDURE SetCoordinates(VAR Peg : TypePeg;
                         VAR X, Y : Integer);
BEGIN
  CASE Peg.PegNumber OF
    1 : X := 11;
    2 : X := 27;
    3 : X := 33
  END;
  Y := (MaxItems - NumberInStack(Peg.Stack))
END;

PROCEDURE EraseDisk(Peg : TypePeg);
VAR
  Row, Column : Integer;
BEGIN
  Row := (BaseLevel - NumberInStack(Peg.Stack) - 1);
  Column := (1 + ((Peg.PegNumber - 1) * 26));
  GotoXY(Column, Row);
```

```
   WriteRepeatedChars(' ', 11);
   Write(' ');
   WriteRepeatedChars(' ', 11)
END;

PROCEDURE ReWriteDisk(Peg : TypePeg;
                      Item : TypeItem);
VAR
   Row, Column : Integer;
BEGIN
   TextColor(LightCyan);
   Row := (BaseLevel - NumberInStack(Peg.Stack));
   Column := (11 + ((Peg.PegNumber - 1) * 26));
   GotoXY(Column - Item, Row);
   WriteRepeatedChars(Disk, 1 + (Item * 2));
   TextColor(LightGray)
END;

PROCEDURE MoveDisks(VAR StartPeg, AuxiliaryPeg,
                        EndPeg : TypePeg;
                        Disks : Integer);
VAR
   Item : TypeItem;
BEGIN
  IF (Disks > 0) THEN
  BEGIN
     MoveDisks(StartPeg, EndPeg, AuxiliaryPeg, Disks - 1);
     SayMoveFrom(startPeg.PegNumber, EndPeg.PegNumber);
     Pop(startPeg.Stack, Item);
     Push(EndPeg.Stack, Item);
```

```
    Delay(600);
    EraseDisk(StartPeg);
    ReWriteDisk(EndPeg, Item);
    MoveDisks(AuxiliaryPeg, StartPeg, EndPeg, (Disks - 1))
  END
END;

PROCEDURE WatchPuzzle(StartPeg, IntermediatePeg,
                      EndPeg : TypePeg);
VAR
  Disks : Integer;
BEGIN
  Disks := NumberInStack(StartPeg.Stack);
  MoveDisks(StartPeg, IntermediatePeg, EndPeg, Disks);
  GotoXY(1, 22); Center('SOLVED !!  Please Press Any Key.');
  Sound(900); Delay(100); Sound(500); Delay(200);
  Sound(300); Delay(100); NoSound;
  REPEAT UNTIL KeyPressed
END;

PROCEDURE GetMove(VAR SourcePeg, TargetPeg : Integer;
                  VAR Quit : Boolean);

  PROCEDURE GetSourcePeg(VAR SourcePeg : Integer;
                         VAR Quit : Boolean);
  VAR
  Choice : Char;
BEGIN
  Quit := False;
```

```
  REPEAT
    GotoXY(1, 22); ClrEol;  {clear lower row}
    GotoXY(1, 21); ClrEol;
    Center('Move FROM PEG 1, 2, 3 or Q (Quit) : ');
    Read(Choice);
    CASE Choice OF
      '1','2','3': SourcePeg := (Ord(Choice) - Ord('1') + 1);
      'Q', 'q' : Quit := True
    END
  UNTIL Choice in ['1', '2', '3', 'Q', 'q']
END;

PROCEDURE GetTargetPeg(VAR TargetPeg : Integer;
                       VAR Quit : Boolean);
VAR
  Choice : Char;
BEGIN
  REPEAT
    GotoXY(1, 22); ClrEol;
    Center('          Move TO PEG 1, 2, or 3  : ');
    Read(Choice);
    CASE Choice OF
      '1','2','3': TargetPeg := (Ord(Choice) - Ord('1') + 1);
      'Q', 'q'   : Quit := True
    END
  UNTIL Choice in ['1', '2', '3', 'Q', 'q']
END;
```

```
BEGIN {GetMove}
   GetSourcePeg(SourcePeg, Quit);
   IF NOT Quit THEN GetTargetPeg(TargetPeg, Quit)
END;

PROCEDURE RemoveDisk(VAR StartPeg, IntermediatePeg,
                         EndPeg : Typepeg;
                         SourcePeg : Integer;
VAR Item : TypeItem);
BEGIN
   CASE SourcePeg OF
      1 : BEGIN
             Pop(StartPeg.Stack, Item);
             EraseDisk(StartPeg)
          END;
      2 : BEGIN
             Pop(IntermediatePeg.Stack, Item);
             EraseDisk(IntermediatePeg)
          END;
      3 : BEGIN
             Pop(EndPeg.Stack, Item);
             EraseDisk(EndPeg)
          END
   END
END;

PROCEDURE AddDisk(VAR StartPeg, IntermediatePeg,
                      EndPeg : TypePeg;
                      TargetPeg : Integer;
                      Item : TypeItem);
```

```
BEGIN
  CASE TargetPeg OF
    1 : BEGIN
          Push(StartPeg.Stack, Item);
          ReWriteDisk(StartPeg, Item)
        END;
    2 : BEGIN
          Push(IntermediatePeg.Stack, Item);
          ReWriteDisk(IntermediatePeg, Item)
        END;
    3 : BEGIN
          Push(EndPeg.Stack, Item);
          ReWriteDisk(EndPeg, Item)
        END;
  END;
END;

FUNCTION MoveOk(StartPeg, IntermediatePeg, EndPeg : TypePeg;
                SourcePeg, TargetPeg : Integer) :Boolean;
VAR
  SourceItem, TargetItem : TypeItem;

  FUNCTION PegIsEmpty( PegNumber : Integer) : Boolean;
  BEGIN
    CASE PegNumber OF
      1 : PegIsEmpty := EmptyStack(StartPeg.Stack);
      2 : PegIsEmpty := EmptyStack(IntermediatePeg.Stack);
      3 : PegIsEmpty := EmptyStack(EndPeg.Stack)
    END
  END;
```

```
BEGIN
  IF (SourcePeg = TargetPeg)
  THEN MoveOk := False
  ELSE IF PegIsEmpty(SourcePeg)
      THEN MoveOk := False
      ELSE IF PegIsEmpty(TargetPeg)
          THEN MoveOk := True
          ELSE
          BEGIN
             RemoveDisk(StartPeg, IntermediatePeg, EndPeg,
                        SourcePeg, SourceItem);
             RemoveDisk(StartPeg, IntermediatePeg, EndPeg,
                        TargetPeg, TargetItem);
             MoveOk := (SourceItem < TargetItem);
             AddDisk(StartPeg, IntermediatePeg, EndPeg,SourcePeg,
                     SourceItem);
             AddDisk(StartPeg, IntermediatePeg, EndPeg,TargetPeg,
                     TargetItem);
          END
END;

PROCEDURE SolvePuzzle( StartPeg, IntermediatePeg,
                       EndPeg : TypePeg);
VAR
  SourcePeg, TargetPeg, Turn : Integer;
  Done : Boolean;
  Item : TypeItem;
BEGIN
  GetMove(SourcePeg, TargetPeg, Done);
  WHILE NOT Done DO
```

```
BEGIN
   IF MoveOk(StartPeg, IntermediatePeg, EndPeg, SourcePeg,
             TargetPeg) THEN
BEGIN
   RemoveDisk(StartPeg, IntermediatePeg, EndPeg, SourcePeg, Item);
   AddDisk(StartPeg, IntermediatePeg, EndPeg, TargetPeg, Item);
   Sound(700); Delay(100); NoSound;  {sound of moving}
   Done := EmptyStack(StartPeg.Stack) AND
           EmptyStack(IntermediatePeg.STack);
   IF Done THEN
   BEGIN
      GotoXY(1, 22); ClrEol;
      GotoXY(1, 21); ClrEol;
      Center('CONGRATULATION !  You Solved It ! ');
      FOR Turn := 1 TO 5 DO
      BEGIN
         Sound(900); Delay(100); Sound(500); Delay(200);
         Sound(300); Delay(100)
      END;
      NoSound;
      REPEAT UNTIL KeyPressed
   END
END
ELSE
BEGIN
   Sound(1000); Delay(100); NoSound;
   GotoXY(1, 20); Center('INVALID Move !'); Delay(1500);
   GotoXY(1, 20); ClrEol
END;
IF NOT Done THEN GetMove(SourcePeg, TargetPeg, Done)
```

```
  END
END;

PROCEDURE DoOption(StartPeg, IntermediatePeg,
                   EndPeg : TypePeg;
                   Choice : Char);
BEGIN
  CASE Choice OF
    'W','w' : WatchPuzzle(StartPeg, IntermediatePeg, EndPeg);
    'S','s' : SolvePuzzle(StartPeg, IntermediatePeg, EndPeg);
  END
END;

PROCEDURE PlayGame( Choice : Char);
VAR
  StartPeg, IntermediatePeg, EndPeg : TypePeg;
BEGIN
  IF Choice <> Quit THEN
  BEGIN
    SetUp(StartPeg, IntermediatePeg, EndPeg);
    DoOption(StartPeg, IntermediatePeg, EndPeg, Choice)
  END
END;

BEGIN {TowersOfHanoi}
  REPEAT
    ShowMenuAndGetChoice(Choice);
    PlayGame(Choice)
  UNTIL (Choice = Quit)
END.
```

11.7.8 เกมจับคู่ภาพ

1. โหมดของจอภาพ กราฟิกส์

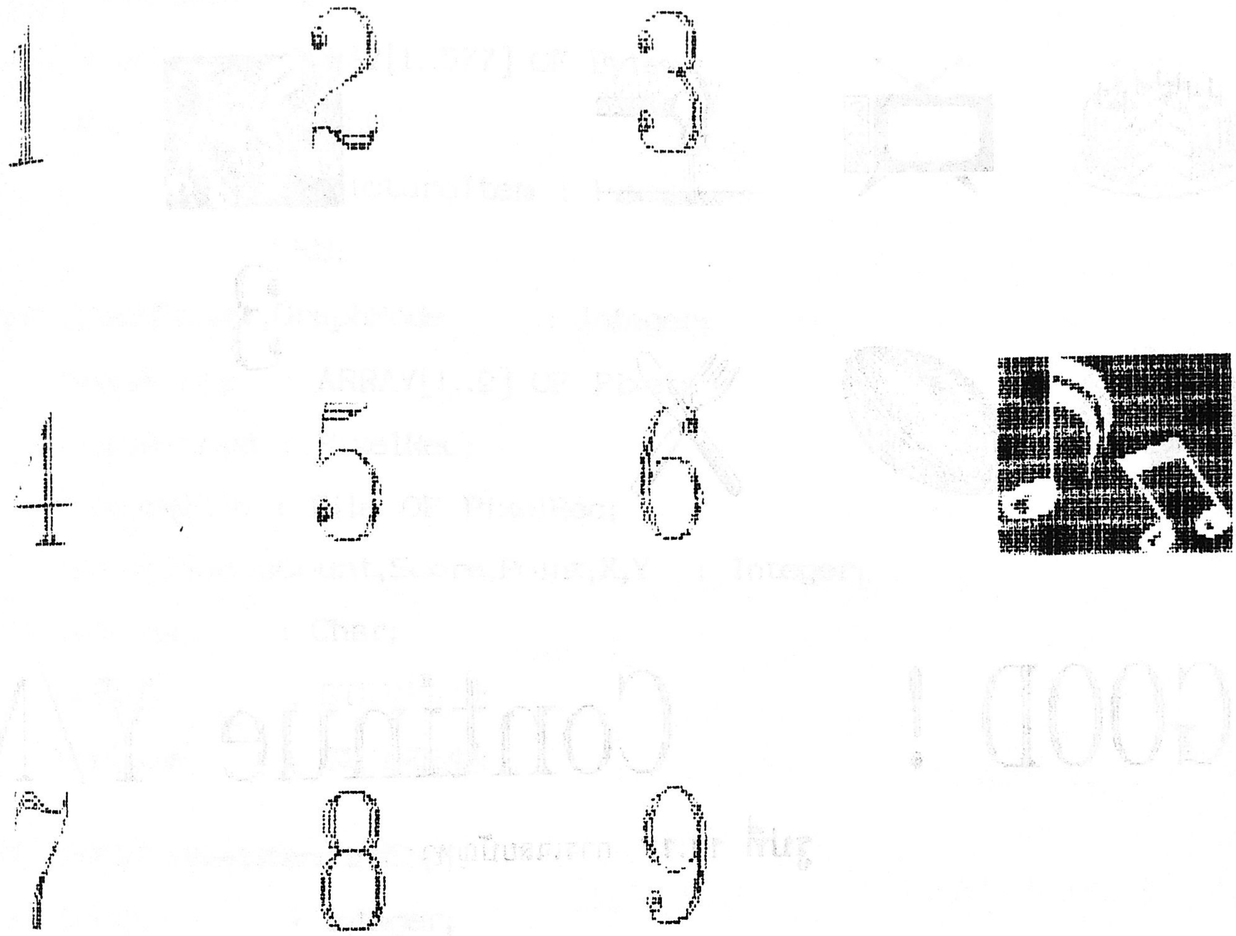
2. กติกา เกมนี้จะต้องใช้ภาพจากไฟล์ภาพ STANDARD.SHP ของโปรแกรมพรินต์มาสเตอร์พลัสด้วย โปรแกรมจะเลือกภาพด้วยวิธีการสุ่มจากภาพทั้งหมด 111 ภาพเพื่อให้ได้ภาพ 9 ภาพ นำภาพที่ได้มาแสดงที่จอภาพเรียงตามลำดับจาก 1 ถึง 9 เมื่อครบทุกภาพแล้วจะปิดภาพเหล่านั้นไว้ด้วยหมายเลขของภาพคือ เลข 1 ถึง 9 และแสดงภาพแบบกลับสีไว้ 1 ภาพ ซึ่งเป็นภาพที่ได้จากการสุ่มมาจากภาพ 9 ภาพที่ปิดเอาไว้ ต่อจากนั้นจึงให้ผู้เล่นทายว่าภาพที่แสดงอยู่นี้ตรงกับภาพหมายเลขใด ถ้าตอบถูกตั้งแต่ครั้งแรกจะได้ 3 คะแนน ถ้าตอบไม่ถูกคอมพิวเตอร์จะเปิดให้ดูว่าภาพที่ทายไม่ถูกนั้นเป็นภาพอะไร แล้วปิดไว้เหมือนเดิม ผู้ทายจะเริ่มทายใหม่ ถ้าทายถูกในครั้งที่ 2 จะได้ 2 คะแนน แต่ถ้าทายถูกในครั้งที่ 3 จะได้คะแนนเพียง 1 คะแนน ถ้าทายไม่ถูกทั้ง 3 ครั้งหรือหลังจากทายถูกแล้ว คอมพิวเตอร์จะแสดงภาพทั้งหมดให้ดูพร้อมกับเฉลยและแสดงคะแนนรวมที่ได้จากการเล่นทั้งหมด

3. วิธีเล่น โปรแกรมเริ่มด้วยคอมพิวเตอร์แสดงภาพ 9 ภาพเรียงตามลำดับดังตัวอย่างในรูปที่ 11.15



รูปที่ 11.15 ภาพที่คอมพิวเตอร์เลือกมา 9 ภาพ

คอมพิวเตอร์จะปิดภาพทั้งหมดทันทีที่แสดงครบ 9 ภาพ ด้วยหมายเลขของภาพนั้นเรียงตามลำดับจาก 1 ถึง 9 แล้วคอมพิวเตอร์จะแสดงภาพกลับสีเป็นภาพคำถามดังตัวอย่างในรูปที่ 11.16



รูปที่ 11.16 ภาพที่คอมพิวเตอร์เลือกให้เป็นคำถาม

หลังจากทายถูกแล้ว หรือทายไม่ถูกเลยทั้ง 3 ครั้ง คอมพิวเตอร์จะเฉลยและแสดงคะแนน ดังตัวอย่างในรูปที่ 11.17



รูปที่ 11.17 การเฉลยปัญหา

4. ข้อสังเกต เราจะต้องติดตามดูภาพตั้งแต่ภาพแรกที่คอมพิวเตอร์แสดง มิ-ฉะนั้นจะดูไม่ทัน โดยเฉพาะภาพสุดท้ายซึ่งมีเวลาแสดงน้อยที่สุด ถ้าเป็นไปได้ควรคิดไว้ด้วยว่าเมื่อกลับสีภาพนั้นน่าจะมีลักษณะอย่างไร

5. โปรแกรม รายละเอียดของโปรแกรมเกมจับคู่ภาพแสดงในโปรแกรม 11.8

โปรแกรม 11.8

```
PROGRAM RandomPictureFromStandardShp;
USES Crt,Graph;
CONST MaxGuess = 3;
TYPE Pixel     = ARRAY[1..577] OF Byte;
     PixelRec = RECORD
                  PictureItem : Pixel;
                END;
VAR GraphDriver,GraphMode       : Integer;
    PixelArray   : ARRAY[1..9] OF Pixel;
    PixelRecord : PixelRec;
    PictureFile : File OF PixelRec;
    Choice,GuessCount,Score,Point,X,Y   : Integer;
    Continue      : Char;
    NoStr          : STRING[1];
    NoScore        : STRING[4];

PROCEDURE ReadStandardShp;
VAR Count          : Integer;
    PicNo          : 1..111;
BEGIN
   FOR Count := 1 TO 9 DO
   BEGIN
      PicNo := Random(112);
      Reset(PictureFile);
      Seek(PictureFile,PicNo-1);
      Read(PictureFile,PixelRecord);
      PixelArray[Count] := PixelRecord.PictureItem;
   END;
END;
```

```
PROCEDURE PicPositionX(PicNo : Integer);
BEGIN
   CASE PicNo OF
      1,4,7  : X := 0;
      2,5,8  : X := 120;
      3,6,9  : X := 240;
   END;
END;

PROCEDURE PicPositionY(PicNo : Integer);
BEGIN
   CASE PicNo OF
      1,2,3  : Y := 0;
      4,5,6  : Y := 100;
      7,8,9  : Y := 200;
   END;
END;

PROCEDURE HidePic(PicNo : Integer);
BEGIN
   SetFillStyle(EmptyFill,PicNo);
   PicPositionX(PicNo);
   PicPositionY(PicNo);
   IF GraphDriver <> 7 THEN Bar(X-8,Y, X+88,Y+52) ELSE
   Bar(X+8,Y,X+8+88,Y+52);
END;

PROCEDURE DisplayNumber(PicNo : Integer);
{VAR  NoStr : STRING[1];}
```

```
BEGIN
   Str(PicNo,NoStr);
   SetTextStyle(TriplexFont,HorizDir,6);
   OutTextXY(X+30,Y,NoStr);
END;

PROCEDURE DisplayPicture(X,Y,PicNo,Reverse : Integer);
VAR ByteNo,ByteInLine,StartX    : Integer;
BEGIN
   StartX := X; ByteInLine := 1;
   FOR ByteNo := 5 TO 576 DO
   BEGIN
      IF Reverse = 1 THEN  {reverse}
      SetLineStyle(UserBitLn,255-PixelArray[PicNo,ByteNo],1)
      ELSE                 {normal}
      SetLineStyle(UserBitLn,PixelArray[PicNo,ByteNo],1);
      IF GraphDriver {VGA} <> 7 THEN Line(X,Y X-15,Y) ELSE {Herc}
      Line(X,Y, X+15,Y); Inc(X,8); Inc(ByteInLine);
      IF ByteInLine > 11 THEN  {number of byte in a line}
      BEGIN
         ByteInLine := 1;
         X := StartX; Inc(Y)
      END
   END
END;

PROCEDURE DisplayAllPic;
VAR Count : Integer;
BEGIN
   FOR Count := 1 TO 9 DO  {display picture}
```

```
  BEGIN
    PicPositionX(Count);
    PicPositionY(Count);
    DisplayPicture(X,Y,Count,0):
  END;
END;

PROCEDURE StartGame;
VAR Count,GuessNo  : Integer;
  Ch          : Char;
BEGIN
  ClearDevice;
  DisplayAllPic;
  {hide picture & display number}
  FOR Count := 1 TO 9 DO
  BEGIN
    HidePic(Count);
    DisplayNumber(Count);
  END;
  REPEAT Choice := Random(10); UNTIL Choice IN[1..9];
  DisplayPicture(450,100,Choice,1);
  GuessCount := 0;   Point := MaxGuess;
  REPEAT
    Inc(GuessCount);
    REPEAT              {1..9 only}
      Ch := ReadKey;
      Val(Ch,GuessNo,Count);
      IF NOT(GuessNo IN[1..9]) THEN Write(#7)
    UNTIL GuessNo IN[1..9];
    IF GuessNo = Choice THEN      {right guess}
```

```
  BEGIN
     Str(Choice,NoStr);
     OutTextXY(490,160,NoStr);
     OutTextXY(20,300,'GOOD !');
     FOR Count := 1 TO 5 DO
     BEGIN
        Sound(900);Delay(150);Sound(400);Delay(100);
        Sound(200);Delay(150);Sound(700);Delay(100);
     END;
     Sound(900); Delay(400); NoSound;
  END
  ELSE        {wrong guess}
  BEGIN
     Sound(800);Delay(80);Sound(200);Delay(100);NoSound;
     PicPositionX(GuessNo);
     PicPositionY(GuessNo);
     HidePic(GuessNo);
     DisplayPicture(X,Y,GuessNo,0); Delay(100);
     HidePic(GuessNo);
     DisplayNumber(GuessNo);      Dec(Point); {point-1}
  END;
UNTIL (GuessCount = MaxGuess) OR (GuessNo = Choice);
IF GuessNo <> Choice THEN OutTextXY(20,300,'TRY ?');
Score := Score + Point;
Str(Choice,NoStr);
OutTextXY(490,160,NoStr);
Str(Score,NoScore);
OutTextXY(480,24,NoScore);
SetTextStyle(TriplexFont,HorizDir,1);
OutTextXY(470,5,'SCORE');
```

```
  SetTextStyle(TriplexFont,HorizDir,6);
  FOR Count := 1 TO 9 DO HidePic(Count);
  DisplayAllPic;
  IF GraphDriver <> 7 THEN
  OutTextXY(200,300,'Continue Y/N') ELSE
  OutTextXY(300,300,'Continue Y/N');
  Continue := ReadKey;
END;

BEGIN
  GraphDriver := Detect;
  InitGraph(GraphDriver, GraphMode, '');
  SetTextStyle(TriplexFont,HorizDir,6);
  OutTextXY(80,100,'PICTURE MATCHING');
  SetTextStyle(TriplexFont,HorizDir,2);
  OutTextXY(200,200,'Press ENTER to Start');
  ReadLn;  Score := 0; {start score}
  Assign(PictureFile,'STANDARD.SHP');
  REPEAT
    Randomize;
    ReadStandardShp;
    StartGame;
  UNTIL Continue IN['n','N'];
  CloseGraph
END.
```

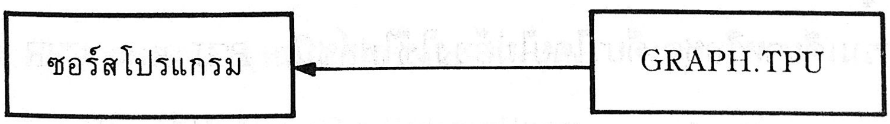
ภาคผนวก ก.
การสร้างโปรแกรมกราฟิกส์เพื่อการเผยแพร่

โปรแกรมกราฟิกส์ที่สร้างขึ้นเพื่อการเผยแพร่สู่สาธารณะ จำเป็นจะต้องจัดให้อยู่ในลักษณะที่สามารถนำไปใช้ได้สะดวก โดยผู้ใช้โปรแกรมไม่จำเป็นจะต้องมีความรู้ใดๆ เกี่ยวกับเทอร์โบปาสคาล

ก.1 การรันโปรแกรมกราฟิกส์จากเทอร์โบปาสคาล

เมื่อพิจารณาวิธีการสร้างโปรแกรมกราฟิกส์ด้วยเทอร์โบปาสคาล จะมีข้อสังเกตดังต่อไปนี้

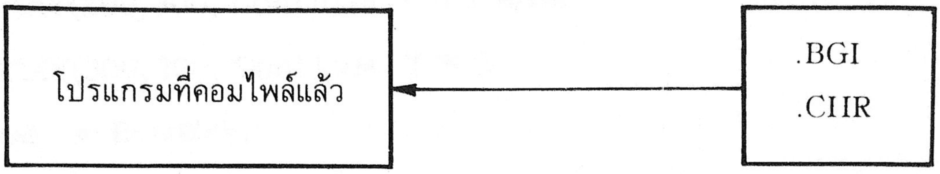
1. ตอนคอมไพล์โปรแกรมจะต้องใช้ไฟล์ GRAPH.TPU ดังแสดงในรูปที่ ก.1



รูปที่ ก.1 ตอนคอมไพล์โปรแกรมมีการเรียกใช้ไฟล์ GRAPH.TPU

2. ตอนรันโปรแกรมจะพบว่าทุกครั้งที่มีการเปลี่ยนโหมดจากเท็กซ์ให้เป็นกราฟิกส์ โปรแกรมจะหาไฟล์ไดรเวอร์ ซึ่งเป็นไฟล์ชนิด BGI มาจากดิสก์ ถ้าไม่มีไฟล์ไดรเวอร์ที่เหมาะสม โปรแกรมจะหยุดทำงาน

ในทำนองเดียวกัน ถ้าในโปรแกรมมีการใช้อักษรกราฟิกส์จากไฟล์ชนิด CHR โปรแกรมก็จะหาไฟล์อักษรชนิดนั้นจากดิสก์ ถ้าหาไม่พบโปรแกรมจะแสดงอักษรตามแบบที่มีอยู่ ซึ่งอาจจะมีผลให้อักษรผิดเพี้ยนไปจากเดิม



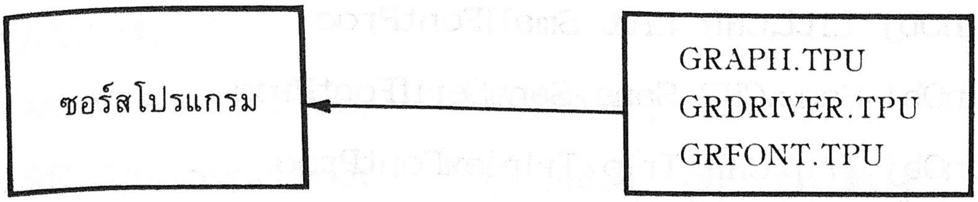
รูปที่ ก.2 ตอนรันโปรแกรมมีการเรียกใช้ไฟล์ BGI และ/หรือ CHR

เพื่อป้องกันการเกิดความผิดพลาดดังกล่าว ผู้สร้างโปรแกรมกราฟิกส์จะต้องจัดเตรียมไฟล์ชนิด BGI และ CHR ทุกไฟล์ไว้ให้แก่ผู้ใช้โปรแกรมด้วย ซึ่งในทางปฏิบัติอาจจะทำให้เกิดความไม่เหมาะสมบางประการขึ้นได้ เช่น กรณีที่เราบันทึกโปรแกรมกราฟิกส์ไว้ในดิสเก็ตโปรแกรมละ 1 แผ่น จะมีไฟล์ BGI และ CHR ซ้ำซ้อนกันในแผ่นดิสก์ทุกแผ่น ทำให้เปลืองเนื้อที่ในดิสก์โดยเปล่าประโยชน์ และสิ่งที่จะเกิดขึ้นอย่างแน่นอนคือ โปรแกรมทำงานช้าเนื่องจากโปรแกรมจะต้องเสียเวลาอ่านไฟล์จากดิสก์

ก.2 หลักการสร้างโปรแกรมกราฟิกส์ที่สะดวกต่อการใช้

ปัญหาในข้อ ก.1 สามารถป้องกันได้โดยการรวมไฟล์ชนิด BGI และ CHR ทั้งหมดเข้าไปอยู่ในโปรแกรมกราฟิกส์ตั้งแต่ตอนคอมไพล์ ซึ่งจะทำให้มีโปรแกรมกราฟิกส์เพียงโปรแกรมเดียวหรือชุดเดียวโดยไม่ต้องใช้ไฟล์ชนิด BGI และ CHR อีกเลย

หลักการรวมไฟล์ชนิด BGI และ CHR เข้าในโปรแกรมกราฟิกส์ได้แก่ การเปลี่ยนไฟล์สองชนิดนี้ให้เป็นยูนิตซึ่งเป็นไฟล์ชนิด TPU เช่น เปลี่ยนไฟล์ BGI ให้เป็นยูนิตชื่อ GRDRIVER.TPU เปลี่ยนไฟล์ CHR ให้เป็นยูนิตชื่อ GRFONT.TPU และเรียกใช้ยูนิตนี้ที่ USES ในโปรแกรม เมื่อเราสั่งคอมไพล์โปรแกรม คอมไพเลอร์จะนำสิ่งต่างๆ ที่เกี่ยวข้อง เช่น คอนสแตนต์ ฟังก์ชัน และโพรซีเยอร์ เข้ามารวมอยู่ในโปรแกรมเช่นเดียวกับการใช้ยูนิต Graph ดังแสดงในรูปที่ ก.3



รูปที่ ก.3 การรวมสิ่งที่มีอยู่ในไฟล์ชนิด TPU เข้าในโปรแกรมตอนทำการคอมไพล์โปรแกรม

เมื่อเรารันโปรแกรมที่ผ่านการคอมไพล์ด้วยวิธีนี้ โปรแกรมจะทำงานได้อย่างสมบูรณ์โดยไม่ต้องอ่านไฟล์ชนิด BGI และ CHR จากดิสก์อีก ดังแสดงในรูปที่ ก.4



รูปที่ ก.4 โปรแกรมที่คอมไพล์โดยรวมยูนิตเอาไว้แล้ว จะสามารถรันได้ทันที

ก.3 การเปลี่ยนไฟล์ชนิด BGI และ CHR ให้เป็นยูนิต

สมมติว่าเราต้องการเปลี่ยนไฟล์ชนิด BGI ให้เป็นไฟล์ GRDRIVER.TPU และเปลี่ยนไฟล์ชนิด CHR ให้เป็น GRFONT.TPU ขั้นตอนดำเนินการมีดังนี้

1. เปลี่ยนไฟล์ชนิด BGI และ CHR ทุกไฟล์ให้เป็นไฟล์ชนิด OBJ โดยใช้โปรแกรม BINOBJ.EXE ซึ่งมีมาพร้อมกับโปรแกรมเทอร์โบปาสคาล วิธีใช้โปรแกรม BINOBJ.EXE ทำได้โดยการพิมพ์คำสั่งที่ DOS ดังนี้

```
BinObj Att.BGI Att ATTDriverProc
BinObj Cga.BGI Cga CgaDriverProc
BinObj EgaVga.BGI EgaVga EgaVgaDriverProc
BinObj Herc.BGI Herc HercDriverProc
BinObj IBM8514.BGI IBM8514 IBM8514DriverProc
BinObj Pc3270.BGI Pc3270 PC3270DriverProc
BinObj Goth.CHR Goth GothicFontProc
```

```
BinObj Litt.CHR Litt SmallFontProc
BinObj Sans.CHR Sans SansSerifFontProc
BinObj Trip.CHR Trip TriplexFontProc
```

หลังจากใช้คำสั่งดังกล่าวนี้จะมีไฟล์ต่างๆ ต่อไปนี้อยู่บนดิสก์

```
Att.OBJ
Cga.OBJ
EgaVga.OBJ
Herc.OBJ
IBM8514.OBJ
Pc3270.OBJ
Goth.OBJ
Litt.OBJ
Sans.OBJ
Trip.OBJ
```

ถึงตอนนี้ไฟล์ที่เคยมีอยู่ 3 ชนิดคือ ไฟล์ BINOBJ.EXE ไฟล์ชนิด BGI และไฟล์ชนิด CHR จะเป็นไฟล์ที่เราไม่จำเป็นต้องใช้อีกต่อไป

2. สร้างยูนิต GrDriver โดยมีขั้นตอนดังนี้

2.1 พิมพ์รายละเอียดของยูนิต GrDriver ดังแสดงในยูนิต 1

ยูนิต 1

```
UNIT GrDriver;
INTERFACE
USES Graph;
```

```
IMPLEMENTATION
PROCEDURE ATTDriverProc;        EXTERNAL; {$L ATT.OBJ}
PROCEDURE CgaDriverProc;        EXTERNAL; {$L CGA.OBJ}
PROCEDURE EgaVgaDriverProc;     EXTERNAL; {$L EGAVGA.OBJ}
PROCEDURE HercDriverProc;       EXTERNAL; {$L HERC.OBJ}
PROCEDURE PC3270DriverProc;     EXTERNAL; {$L PC3270.OBJ}
PROCEDURE IBM8514DriverProc;    EXTERNAL; {$L IBM8514.OBJ}

PROCEDURE Error(S : STRING);
BEGIN
  WriteLn(S,' : ',GraphErrorMsg(GraphResult));
  ReadLn; Halt
END;

BEGIN
  IF RegisterBGIDriver(@EgaVgaDriverProc) < 0
    THEN Error('VGA ');
  IF RegisterBGIDriver(@AttDriverProc) < 0
    THEN Error('Att ');
  IF RegisterBGIDriver(@CgaDriverProc) < 0
    THEN Error('CGA ');
  IF RegisterBGIDriver(@PC3270DriverProc) < 0
    THEN Error('PC3270 ');
  IF RegisterBGIDriver(@IBM8514DriverProc) < 0
    THEN Error('IBM8514 ');
```

```
   IF RegisterBGIDriver(@HercDriverProc)   < 0
      THEN Error('Hercules ');
END.
```

2.2 บันทึก (save) ยูนิต 1 ไว้ในดิสก์โดยใช้ชื่อ GRDRIVER.PAS

2.3 คอมไพล์ยูนิต GrDriver โดยเลือก Destination เป็น Disk เพื่อบันทึกผลการคอมไพล์ไว้ในดิสก์ ซึ่งหลังจากคอมไล้แล้วจะมียูนิต GRDRIVER.TPU อยู่บนดิสก์

3. สร้างยูนิต GrFont โดยมีขั้นตอนดังนี้

3.1 พิมพ์รายละเอียดของยูนิต GrFont ดังแสดงในยูนิต 2

ยูนิต 2

```
UNIT GrFont;
INTERFACE
USES Graph;

IMPLEMENTATION
PROCEDURE GothicFontProc;      EXTERNAL; {$L GOTH.OBJ }
PROCEDURE SansSerifFontProc; EXTERNAL; {$L SANS.OBJ }
PROCEDURE SmallFontProc;       EXTERNAL; {$L LITT.OBJ }
PROCEDURE TriplexFontProc;     EXTERNAL; {$L TRIP.OBJ }

PROCEDURE Error(S : STRING);
BEGIN
```

```
    WriteLn(S,' font : ',GraphErrorMsg(GraphResult));
    ReadLn; Halt
END;

BEGIN
    IF RegisterBGIFont(@GothicFontProc) < 0
       THEN Error('Gothic ');
    IF RegisterBGIFont(@SansSerifFontProc) < 0
       THEN Error('SanSerif ');
    IF RegisterBGIFont(@SmallFontProc) < 0
       THEN Error('Small ');
    IF RegisterBGIFont(@TriplexFontProc) < 0
       THEN Error('Triplex ');
END.
```

3.2 บันทึก (save) ยูนิต 2 ไว้ในดิสก์โดยใช้ชื่อ GRFONT.PAS

3.3 คอมไพล์ยูนิต GrFont โดยเลือก Destination เป็น Disk เพื่อบันทึกผลการคอมไพล์ไว้ในดิสก์ ซึ่งหลังจากคอมไล์แล้วจะมียูนิต GRFONT.TPU อยู่บนดิสก์

ก.4 การใช้ยูนิต GrDriver และ GrFont

ถึงแม้ว่ายูนิตทั้งสองนี้จะแยกกันอยู่ แต่การเรียกใช้จะต้องสัมพันธ์กัน กล่าวคือ เราจะเรียกใช้ยูนิต GrFont ในโปรแกรมหรือไม่ก็ได้ ถ้ามีการเรียกใช้ยูนิตนี้ เราจะต้องจัดให้ยูนิต GrFont อยู่หลังยูนิต GrDriver เสมอ ในทำนองเดียวกัน การใช้ยูนิต GrDriver ก็จะต้องจัดให้อยู่หลังยูนิต Graph ด้วย ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1 USES Graph,GrDriver;

หมายความว่า โปรแกรมนี้สามารถจะแสดงกราฟิกส์ที่จอภาพได้ทุกชนิด แต่สามารถแสดงอักษรกราฟิกส์ได้เพียงแบบเดียวคืออักษรแบบ DefaultFont

ตัวอย่างที่ 2 USES Graph,GrDriver,GrFont;

หมายความว่า โปรแกรมนี้สามารถจะแสดงกราฟิกส์ที่จอภาพได้ทุกชนิด และสามารถแสดงอักษรกราฟิกส์ที่มีในเทอร์โบปาสคาลได้ครบทุกแบบ

โปรแกรมกราฟิกส์ที่สร้างด้วยวิธีการนี้ สามารถรันได้อิสระโดยไม่ต้องอาศัยไดรเวอร์จากภายนอกอีกเลย ทำให้สามารถรันได้อย่างรวดเร็วและสามารถทำงานบนจอภาพได้ทุกชนิด

อย่างไรก็ดี โปรแกรมที่คอมไพล์ให้เป็นไฟล์ EXE ด้วยวิธีนี้จะเป็นไฟล์ที่มีขนาดใหญ่กว่าการคอมไพล์โปรแกรมโดยใช้ยูนิต Graph เพียงยูนิตเดียว วิธีลดขนาดไฟล์อาจทำได้โดยการไม่รวมไดรเวอร์ที่คาดว่าไม่ได้ใช้ เช่น CGA PC3270 เข้าในยูนิต GrDriver ซึ่งทำได้โดยการลบสเตตเมนต์ที่เกี่ยวข้องกับไดรเวอร์เหล่านี้ออกจากยูนิต 1 ก่อนการคอมไพล์

ภาคผนวก ข.

การเตรียมยูนิตเพื่อให้คอมไพล์ได้รวดเร็ว

ข.1 ชนิดของไฟล์ที่เก็บยูนิต

ไฟล์ที่เก็บยูนิตมี 2 ชนิดคือ ชนิด TPU และ TPL เมื่อเราคอมไพล์ยูนิต คอมไพเลอร์จะเก็บยูนิตนั้นไว้ในไฟล์ที่มีชื่อเดียวกันกับยูนิตและกำหนดชนิดของไฟล์เป็น TPU (Turbo Pascal Unit) โดยอัตโนมัติ ซึ่งในไฟล์จะมีเพียงยูนิตเดียว ส่วนไฟล์ชนิด TPL (Turbo Pascal Library) จะมีหลายยูนิตอยู่รวมกัน เช่น ในไฟล์ TURBO.TPL ตามปกติจะมี 5 ยูนิตคือ System Dos Crt Overlay และ Printer ในตอนคอมไพล์โปรแกรมที่เรียกใช้ยูนิต คอมไพเลอร์จะหายูนิตนั้นจาก TURBO.TPL ก่อนถ้าหาไม่พบจะไปหาจากไฟล์ชนิด TPU และถ้าหาไม่พบอีกการคอมไพล์จะผิดพลาดและหยุดคอมไพล์

ข.2 การรวมยูนิต TPU เข้าในไฟล์ TURBO.TPL

การรวมยูนิตในไฟล์ชนิด TPU เข้ากับไฟล์ TURBO.TPL จะให้ทำให้เกิดความสะดวกและรวดเร็วในการคอมไพล์โปรแกรมเพราะยูนิตทั้งหมดที่มีอยู่ในไฟล์ TURBO.TPL จะถูกอ่านเข้าไปเก็บไว้ในหน่วยความจำของคอมพิวเตอร์ตั้งแต่ตอนเริ่มต้นคือ ตั้งแต่ตอนที่เราเรียกโปรแกรมคอมไพเลอร์ TURBO จึงมีผลให้คอมไพเลอร์สามารถหายูนิตได้อย่างรวดเร็ว ดังนั้น ถ้ามียูนิตใดที่ต้องใช้บ่อยเราก็ควรจะรวมยูนิตนั้นไว้ในไฟล์ TURBO.TPL

วิธีการรวมยูนิตหนึ่งเข้ากับ TURBO.TPL ทำได้โดยใช้โปรแกรม TPUMOVER.EXE ซึ่งมีมาให้พร้อมกับเทอร์โบปาสคาล

ตัวอย่าง สมมติว่าเราจะรวม GRDRIVER.TPU ซึ่งอยู่ในดิสก์ B เข้ากับ TURBO.TPL ซึ่งอยู่ในดิสก์ A และมีโปรแกรม TPUMOVER.EXE อยู่ในดิสก์ A ด้วย ขั้นตอน การรวมมีดังนี้

1. ที่ A> พิมพ์ TPUMOVER และกดคีย์ Enter จอภาพจะปรากฏดังรูปที่ ข.1

Turbo Pascal Unit Librarian

Version 5.5

A:\TURBO.TPL

Unit	Code	Data	Syms	Uses
SYSTEM	21659	664	4327	
OVERLAY	1844	24	975	
CRT	1555	20	1721	
DOS	1593	6	4107	
PRINTER	56	256	305	

A:\NONAME.TPU

Unit	Code	Data	Syms	Uses

File size: 43 K

Drive A: 1 K free

รูปที่ ข.1 จอภาพแรกของโปรแกรม TPUMOVER

ที่ด้านล่างของจอภาพแสดงคีย์สำหรับควบคุมดังนี้

F1-Help F2-Save F3-New F4-Info F6-Switch

+-Mark Ins-Copy Del-Delete Esc-Quit

2. กดคีย์ F6 เพื่อไปที่กรอบทางขวา (A:\NONAME.TPU)

3. กดคีย์ F3 เพื่ออ่านไฟล์ TPU ซึ่งโปรแกรมจะให้ป้อนชื่อไฟล์ ในที่นี้ให้ป้อนชื่อไฟล์เป็น B:GRDRIVER.TPU ต่อจากนั้นจะมีการอ่านไฟล์จากดิสก์ B ซึ่งที่จอภาพจะเป็นดังรูปที่ ข.2

Turbo Pascal Unit Librarian

Version 5.5

A:\TURBO.TPL

Unit	Code	Data	Syms	Uses
SYSTEM	21659	664	4327	
OVERLAY	1844	24	975	
CRT	1555	20	1721	
DOS	1593	6	4107	
PRINTER	56	256	305	

File size: 43 K

Drive A: 1 K free

B:\GRDRIVER.TPU

Unit	Code	Data	Syms	Uses
GRDRIVER	37031	0	963	GRAPH

File size: 37 K

Drive B: 238 K free

รูปที่ ข.2 ผลจากการอ่านไฟล์ GRDRIVER.TPU เข้ามาในหน่วยความจำ

4. กดคีย์ + เพื่อกำหนดชื่อยูนิตที่ต้องการ ซึ่งก็คือ GRDRIVER.TPU

5. กดคีย์ Ins เพื่อก๊อบปี้ยูนิตนั้นไปที่กรอบทางซ้าย (A:\TURBO.TPL) ซึ่งจะมีชื่อ GRDRIVER.TPU ปรากฏที่ด้านล่างของกรอบทางซ้าย

6. กดคีย์ F6 เพื่อไปที่กรอบทางซ้าย (A:\TURBO.TPL)

7. กดคีย์ F2 เพื่อบันทึกยูนิตในกรอบทางซ้ายไว้ในไฟล์ TURBO.TPL

ถึงตอนนี้ยูนิตชื่อ GrDriver ก็ได้เข้ามาเป็นส่วนหนึ่งของ TURBO.TPL แล้ว ถ้าต้องการรวมยูนิตอื่นๆ อีก (เช่น GRFONT.TPU) หลังจากกดคีย์ F2 ตามข้อ 7 แล้วให้กดคีย์ F6 เพื่อไปกรอบทางขวา กดคีย์ F3 เพื่อเลือกไฟล์ใหม่ ต่อจากนั้นจึงดำเนินการซ้ำตามวิธีเดิมจากข้อ 4 ถึง 7 เมื่อต้องการเลิกงานจึงกดคีย์ Esc

ข.3 การลบยูนิต TPU ออกจากไฟล์ TURBO.TPL

การลบยูนิตออกจาก TURBO.TPL จะมีผลให้เทอร์โบปาสคาลใช้หน่วยความจำน้อยลง เช่น อาจจะลบยูนิต Overlay ออกไป ถ้าเราไม่ได้ใช้ยูนิตนี้

ตัวอย่าง วิธีการลบยูนิต Overlay จากไฟล์ TURBO.TPL ซึ่งอยู่ในดิสก์ A และโปรแกรม TPUMOVER อยู่ในดิสก์ A มีขั้นตอนดังนี้

1. ที่ A> พิมพ์ TPUMOVER และกดคีย์ Enter
2. กดคีย์ลูกศรขึ้นหรือลงเพื่อเลื่อนแถบสว่างไปที่ OVERLAY
3. กดคีย์ + เพื่อเลือก OVERLAY
4. กดคีย์ Del เพื่อลบ OVERLAY
5. กดคีย์ F2 เพื่อบันทึกยูนิตที่เหลือไว้ใน TURBO.TPL
6. กดคีย์ Esc เพื่อเลิกใช้ TPUMOVER

ภาคผนวก ค.

รหัส ASCII

ตารางที่ ค.1 รหัส ASCII

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
0	0		16	10	►	32	20		48	30	0
1	1	☺	17	11	◄	33	21	!	49	31	1
2	2	☻	18	12	↕	34	22	"	50	32	2
3	3	♥	19	13	!!	35	23	#	51	33	3
4	4	♦	20	14	¶	36	24	$	52	34	4
5	5	♣	21	15	§	37	25	%	53	35	5
6	6	♠	22	16	■	38	26	&	54	36	6
7	7	•	23	17	↨	39	27	'	55	37	7
8	8	◘	24	18	↑	40	28	(	56	38	8
9	9	○	25	19	↓	41	29	)	57	39	9
10	A	◙	26	1A	→	42	2A	*	58	3A	:
11	B	♂	27	1B	←	43	2B	+	59	3B	;
12	C	♀	28	1C	∟	44	2C	,	60	3C	<
13	D	♪	29	1D	↔	45	2D	–	61	3D	=
14	E	♫	30	1E	▲	46	2E	.	62	3E	>
15	F	¤	31	1F	▼	47	2F	/	63	3F	?

ตารางที่ ค.1 (ต่อ) รหัส ASCII

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
64	40	@	90	5A	Z	116	74	t	142	8E	Ä
65	41	A	91	5B	[	117	75	u	143	8F	Å
66	42	B	92	5C	\	118	76	v	144	90	É
67	43	C	93	5D	]	119	77	w	145	91	æ
68	44	D	94	5E	^	120	78	x	146	92	Æ
69	45	E	95	5F	_	121	79	y	147	93	ô
70	46	F	96	60		122	7A	z	148	94	ö
71	47	G	97	61	a	123	7B	{	149	95	ò
72	48	H	98	62	b	124	7C	|	150	96	û
73	49	I	99	63	c	125	7D	}	151	97	ù
74	4A	J	100	64	d	126	7E	~	152	98	ÿ
75	4B	K	101	65	e	127	7F	Δ	153	99	Ö
76	4C	L	102	66	f	128	80	Ç	154	9A	Ü
77	4D	M	103	67	g	129	81	ü	155	9B	¢
78	4E	N	104	68	h	130	82	é	156	9C	£
79	4F	O	105	69	i	131	83	â	157	9D	¥
80	50	P	106	6A	j	132	84	ä	158	9E	Pt
81	51	Q	107	6B	k	133	85	à	159	9F	*f*
82	52	R	108	6C	l	134	86	å	160	A0	á
83	53	S	109	6D	m	135	87	ç	161	A1	í
84	54	T	110	6E	n	136	88	ê	162	A2	ó
85	55	U	111	6F	o	137	89	ë	163	A3	ú
86	56	V	112	70	p	138	8A	è	164	A4	ñ
87	57	W	113	71	q	139	8B	ï	165	A5	Ñ
88	58	X	114	72	r	140	8C	î	166	A6	ª
89	59	Y	115	73	s	141	8D	ì	167	A7	º

ตารางที่ ค.1 (ต่อ) รหัส ASCII

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
168	A8	¿	190	BE	╛	212	D4	╘	234	EA	Ω
169	A9	⌐	191	BF	┐	213	D5	╒	235	EB	δ
170	AA	¬	192	C0	└	214	D6	╓	236	EC	∞
171	AB	½	193	C1	┴	215	D7	╫	237	ED	ø
172	AC	¼	194	C2	┬	216	D8	╪	238	EE	∈
173	AD	!	195	C3	├	217	D9	┘	239	EF	∩
174	AE	«	196	C4	─	218	DA	┌	240	F0	≡
175	AF	»	197	C5	┼	219	DB	█	241	F1	±
176	B0	░	198	C6	╞	220	DC	▄	242	F2	≥
177	B1	▒	199	C7	╟	221	DD	▌	243	F3	≤
178	B2	▓	200	C8	╚	222	DE	▐	244	F4	⌠
179	B3	│	201	C9	╔	223	DF	▀	245	F5	⌡
180	B4	┤	202	CA	╩	224	E0	α	246	F6	÷
181	B5	╡	203	CB	╦	225	E1	β	247	F7	≈
182	B6	╢	204	CC	╠	226	E2	ϒ	248	F8	.
183	B7	╖	205	CD	═	227	E3	Π	249	F9	•
184	B8	╕	206	CE	╬	228	E4	Σ	250	FA	.
185	B9	╣	207	CF	╧	229	E5	σ	251	FB	√
186	BA	║	208	D0	╨	230	E6	μ	252	FC	^{n}
187	BB	╗	209	D1	╤	231	E7	τ	253	FD	^{2}
188	BC	╝	210	D2	╥	232	E8	Φ	254	FE	•
189	BD	╜	211	D3	╙	233	E9	θ	255	FF	

ภาคผนวก ง.

รหัสคีย์เพิ่มเติม

เมื่อคีย์ถูกกดคอมพิวเตอร์จะได้รับรหัสเป็นรหัส ASCII ซึ่งมีค่าและความหมายดังแสดงในภาคผนวก ค. สำหรับคีย์ที่ไม่มีในตาราง ASCII เช่น ฟังก์ชันคีย์ (F1-F12) คีย์ลูกศร Alt, Ctrl ซึ่งส่วนใหญ่จะใช้เป็นคีย์ควบคุม เมื่อคีย์หรือกลุ่มคีย์ดังกล่าวนี้ถูกกดคอมพิวเตอร์จะได้รับรหัส 2 ค่า โดยค่าแรกจะเป็น 0 และค่าที่สองจะมีความหมายแทนคีย์ต่างๆ ดังแสดงในตารางที่ ง.1 ต่อไปนี้

วิธีนำค่าจากคีย์เหล่านี้ไปใช้งานทำได้โดยการตรวจสอบรหัสค่าแรก ถ้าไม่เป็น 0 แสดงว่าเป็นรหัส ASCII ปกติ ถ้ารหัสแรกเป็น 0 แสดงว่าเป็นคีย์เพิ่มเติม ให้ตรวจสอบต่อไปว่ารหัสตัวที่ 2 มีค่าเป็นอะไร แล้วจึงนำค่านั้นไปใช้เป็นเงื่อนไขในโปรแกรมต่อไป

ตัวอย่างโปรแกรมตรวจสอบคีย์เป็นดังนี้

```
PROGRAM TestKey;
USES Crt;
VAR Ch : Char;
BEGIN
   Ch := ReadKey;
```

```
    IF Ch = #0 THEN
    WriteLn('Extended key code : ', Ord(ReadKey))
              ELSE WriteLn('Normal key : ',Ch);
    ReadLn
END.
```

ตารางที่ ง.1 รหัสค่าที่สองและคีย์ที่ถูกกด

รหัส	คีย์ที่กด
3	NUL
15	Shift Tab
16–25	Alt Q,W,E,R,T,Y,U,I,O,P
30–38	Alt A,S,D,F,G,H,J,K,L
44–50	Alt Z,X,C,V,B,N,M
59–68	F1 to F10
71	Home
72	Up Arrow
73	PgUp
75	Left Arrow
77	Right Arrow
79	End
80	Down Arrow
81	PgDn
82	Ins
83	Del
84–93	Shift F1 to Shift F10

ตารางที่ ง.1 (ต่อ) รหัสค่าที่สองและคีย์ที่ถูกกด

รหัส	คีย์ที่กด
94-103	Ctrl F1 to Ctrl F10
104-113	Alt F1 to Alt F10
114	Ctrl PrtSc
115	Ctrl Left Arrow
116	Ctrl Right Arrow
117	Ctrl End
118	Ctrl PgDn
119	Ctrl Home
120-131	Alt 1,2,3,4,5,6,7,8,9,0,-,=
132	Ctrl PgUp
133	F11
134	F12
135	Shift F11
136	Shift F12
137	Ctrl F11
138	Ctrl F12
139	Alt F11
140	Alt F12

หนังสืออ้างอิง

Borland international. <u>TURBO PASCAL OWNER'S HANDBOOK.</u> USA : Borland international, 1987.

Borland international. <u>TURBO PASCAL PROFESSIONAL 5.5.</u> USA : Borland international, 1990.

Borland international. <u>TURBO PASCAL 6.0.</u> USA : Borland International, 1990.

Dewey, Bruce R. <u>COMPUTER GRAPHICS FOR ENGINEERS.</u> USA : Harper & Row, Publisher, Inc., 1988.

Duntemann, Jeff. <u>COMPLETE TURBO PASCAL.</u> 3rd ed. USA : Scott, Foresman and Company, 1989.

Goldstein, Lary Joel. <u>HANDS-ON TURBO PASCAL.</u> USA : Simon & Schuster, Inc., 1991.

Hill, F.S. Jr. COMPUTER GRAPHICS. USA : Macmillan Publishing Company, 1990.

Jamsa, Kris and Steven Nameoff. TURBO PASCAL PROGRAMMER'S LIBRARY. 2nd ed. USA : Osborne McGraw-Hill, 1988.

Kingslake, Richard. AN INTRODUCTORY COURSE IN COMPUTER GRAPGHICS. Sweden : Studentlitteratur, Lund, 1986.

Mitchell, William J. and The Others. THE ART OF COMPUTER GRAPHICS PROGRAMMING. USA : Von Nostrand Reinhold Company, Inc.,1987.

Rankin, John R. COMPUTER GRAPHICS SOFTWARE CONSTRUCTION. Australia : Prentice Hall Australia Pty Ltd, 1989.

Schildt, Herbert. ADVANCED TURBO PASCAL PROGRAMMING & TECHNIQUES. USA : Osborne McGraw-Hill, 1987.

Soybel, Jeremy G. TURBO PASCAL 5.5 PROGRAMMING. USA : Windcrest Books, 1990.

หนังสือเล่มนี้สำหรับ	
☑ ผู้เริ่มต้น ☑ ระดับกลาง	☐ ระดับสูง
☑ เรียนรู้การใช้	☐ คู่มืออ้างอิง
☑ IBM Compat.	☐ Macintosh
PROGRAMMING LANGUAGE	

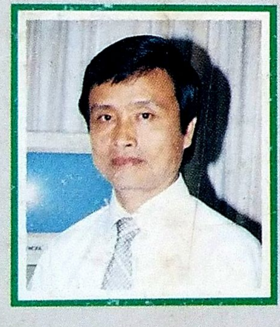
วิธีที่ดีที่สุดสำหรับการสร้างภาพกราฟิกส์ด้วยคอมพิวเตอร์ก็คือการเขียนโปรแกรมเพราะเป็นวิธีการเดียวที่เราสามารถจะสร้างภาพกราฟิกส์ได้ทุกชนิดโดยไม่มีขีดจำกัด

หนังสือเล่มนี้มีรายละเอียดเกี่ยวกับ ความรู้พื้นฐานทางด้านคอมพิวเตอร์กราฟิกส์ การสร้างภาพกราฟิกส์ชนิดต่าง ๆ การให้สีและลายแก่ภาพ การแปลงภาพกราฟิกส์ การพิมพ์ภาพกราฟิกส์ในรูปแบบต่าง ๆ การสร้างอักษรกราฟิกส์ การบันทึกภาพกราฟิกส์ การนำภาพจากโปรแกรมอื่นมาใช้งาน

เพื่อให้เห็นตัวอย่างการนำโปรแกรมคอมพิวเตอร์กราฟิกส์มาใช้งาน จึงได้กล่าวถึงการสร้างและพิมพ์กราฟจากข้อมูลที่กำหนดให้ ซึ่งสามารถสร้างกราฟได้ 4 ชนิด บทสุดท้ายเป็นเรื่องของการสร้างเกมคอมพิวเตอร์ ซึ่งมีตัวอย่างโปรแกรมเกมรวม 8 เกม

ทุกหัวข้อในหนังสือนี้มีการอธิบายอย่างละเอียดเป็นขั้นตอนตามลำดับ จากง่ายไปหายาก เพื่อให้ผู้อ่านสามารถทำความเข้าใจได้เร็วที่สุด และเสริมประสบการณ์ด้วยโปรแกรมตัวอย่างซึ่งต้องใช้โปรแกรมถึง 157 โปรแกรมและ 2 ยูนิต ทุกโปรแกรมทดสอบแล้วด้วยคอมพิวเตอร์ทั้งระบบจอภาพสีวีจีเอและจอภาพเฮอร์คิวลีส

โปรแกรมทั้งหมดใช้ได้กับเทอร์โบปาสคาล เวอร์ชัน 4.0, 5.0, 5.5, 6.0 และเวอร์ชันล่าสุด



นุกูล กระจาย ได้รับปริญญาศิลปศาสตร์มหาบัณฑิต จากคณะเศรษฐศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ และได้รับวุฒิบัตร Microprocessors Technology and Applications in Physics. จาก ICTP ประเทศอิตาลี ปัจจุบันเป็นอาจารย์สังกัดคณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ วิทยาเขตประสานมิตร มีประสบการณ์เกี่ยวกับการสอนวิชาคอมพิวเตอร์ทั้งทางด้านโปรแกรมสำเร็จรูปและการเขียนโปรแกรม เป็นผู้ออกแบบและพัฒนาระบบประมวลผลสอบให้แก่มหาวิทยาลัยตั้งแต่ปีพ.ศ. 2526 เคยเป็นผู้ชำนาญสาขาคอมพิวเตอร์ที่สถาบันส่งเสริมการสอนวิทยาศาสตร์และเทคโนโลยี (สสวท.) ดำเนินการเกี่ยวกับการเขียนตำราเรียนและคู่มือครูวิชาคอมพิวเตอร์ในระดับมัธยมปลาย พร้อมทั้งเป็นวิทยากรอบรมครูผู้สอนวิชาคอมพิวเตอร์ในระดับมัธยมปลายทั่วประเทศ เขียนโปรแกรมกับเทอร์โบปาสคาลมาตั้งแต่เวอร์ชัน 2.0 จนถึง 6.0 มีผลงานที่เป็นบทความเกี่ยวกับคอมพิวเตอร์ วิทยาศาสตร์ และเทคโนโลยี ประมาณ 200 เรื่อง มีงานวิจัยเกี่ยวกับคอมพิวเตอร์ซอฟต์แวร์ 1 เรื่อง หนังสือเล่มแรกที่ซีเอ็ดฯ จัดพิมพ์ คือ *การเขียนโปรแกรมและประมวลผลข้อมูลด้วยเทอร์โบปาสคาล* ซึ่งขอเสนอแนะให้เป็นหนังสือระดับพื้นฐานของหนังสือเล่มนี้สำหรับผู้อ่านที่ยังไม่คุ้นเคยกับเทอร์โบปาสคาล

ISBN 974-509-565-6
9 789745 095656
161 **375 บาท**